The Applicability of Hypermedia Technology to Database Design.

Leonard Bonde.

**Department of Computer Studies,
University of Glamorgan, U.K.**

A thesis submitted in partial fulfilment of the requirements of the University of Glamorgan/Prifysgol Morgannwg for the degree of Doctor of Philosophy.

January 1995.

List of Figures.

Chapter 2.

Chapter 3.

Chapter 4.

Chapter 5.

Chapter 6.

Chapter 7.

Chapter 8.

Appendix 2.

List of Tables.

Chapter 7.

Chapter 8.

Acknowledgements.

I would like to record my sincere thanks to Dr. Paul Beynon-Davies who has acted as my Director of Studies throughout the course of this research. His positive support and constructive attitude has ensured completion of this work. I would also like to give my unreserved thanks to my other supervisors, Dr. Chris Jones and Mr. Duncan McPhee for their help and encouragement. My gratitude is extended to Mr. Colin Bowring for his involvement at the commencement of this work.

I am also indebted to the many members of staff and students in the Department of Computer Studies at the University of Glamorgan who have positively supported my research. My special thanks goes to the following members of staff: Mr. Kevin Sewell (for technical support), Dr. Doug Tudhope (for useful comments on a number of aspects of the work), and to the following for taking part in my evaluation sessions: Mr. Tim Hutchings and Mr. Andrew Mortimer, Mr. Carl Taylor and Mr. Graham Bevan, Dr. Rob Davies and Mr. Andy Gray, Dr. Dave Kidner and Dr. Mark Ware, Mr. Mike Watkins and Mrs. Geneen Stubbs, Mr. Jim Moon and Mr. Danny Thomas, Mr. Veasey and Mr. Erwan Larmor.

I would like to express my sincere gratitude to the Overseas Development Administration (ODA) and the University of Glamorgan for jointly awarding me a scholarship that made this research possible.

My family deserves special recognition for being so supportive under trying and difficult circumstances over the past three and a half years.

I dedicate this work to my father (the late Mr. Basil Bonde Chawira) and to my mother (Mrs. Leah Bonde).

Vhuramavi - mhofu yemukono - Chifamba nenhindi yemago - Seke Mutema !

Declaration.

I declare that, with the exception of the assistance acknowledged, this thesis is the result of my own work. It has not been accepted for any other award and is not currently being submitted in candidature for any other award.

Signed: _____

Date : 04-01-95

Abstract.

The Applicability of Hypermedia Technology to Database Design.

Leonard Bonde (University of Glamorgan, U.K.)

This work discusses an investigation into the applicability of hypermedia technology, in particular problem-resolution systems, to the conceptual modelling stage of the database design process. Conceptual modelling produces a representative abstraction (schema) of the data requirements of an organisation. Hypermedia is a form of information representation and management based on the principle of connecting nodes of self-contained chunks of information with links. It therefore seems suited to the representation and management of irregularly-structured design documents.

Conceptual modelling comprises schema modelling and schema integration. The complexity of conducting large-scale conceptual modelling work can be reduced by developing individual schemas of subsets of an organisation's data requirements (schema modelling) and then amalgamating them (schema integration). Schema modelling involves identifying entities whose data is to be represented in a database along with their relationships. Schema modelling can exploit the modularity offered by hypermedia in representing these concepts.

Inevitably, individual schemas may duplicate some data, leading to undesirable redundancy. Schema integration aims at producing a conceptual schema with no redundancy. This process requires detailed information, which is perspective-dependent and is often missing in the subschemas. Thus schema integration is necessarily a process of achieving semantic consensus. Achieving semantic consensus is necessarily a collaborative process which involves negotiation between the various stakeholders in the conceptual schema. The valuable information resulting from such negotiation is often lost because of lack of mechanisms to effectively capture such ill-structured information. The Issue-Based Information System (IBIS) [Rittel and Kunz 1970] provides a framework that supports and documents negotiation. A collaborative hypermedia-based IBIS environment for schema integration may reduce this problem.

It is proposed that the main contributions to knowledge are:

a. The construction of working prototypes that demonstrate many of the features of a possible infrastructure for collaborative conceptual modelling.
b. The assessment of the usefulness of a collaborative schema integration tool in capturing integration rationale by consensus.

Chapter 1

Applying Hypermedia Technology to Collaborative Database Design?

Summary.

This thesis discusses an investigation into the applicability of hypermedia to database design by focusing on a specific phase of the database design process: conceptual modelling. Hypermedia is a form of information representation and management based on the principle of connecting nodes of self-contained chunks of information with links. In summary, conceptual modelling is that phase of the database design process during which data analysts produce a model (known as the conceptual schema) representing the data requirements of an organisation. The conceptual model is independent of implementation details and represents an abstraction of the universe of discourse in terms of entities and their relationships. Further, the properties of entities and relationships are represented in terms of attributes.

Conceptual modelling is comprised of two subphases: schema modelling, followed by schema integration. This division is necessary because of the complex and time-consuming nature of large-scale data modelling work. Much of this difficulty can be overcome by conceiving of an organisation as a collection of interdependent information areas. An information area is defined as a subset of an organisation that has functional closure and is served by well-defined data. Schema modelling is responsible for producing representations of each information area. This obviously results in a number of subschemas for the organisation, which may inevitably overlap with respect to the data space they cover. However, a single conceptual schema has to be produced, and this is the function of the schema integration phase: to amalgamate the individually produced subschemas.

Thus schema modelling involves identifying entities and relationships (which are indeed self-contained chunks of data in their own right) and encoding them as a network of design documents. This seems to make hypermedia a suitable environment to represent and manage schemas.

Schema integration is necessarily a negotiation process between the data analysts responsible for the production of the subschemas. An essential aspect of the negotiation process is the need to reach consensus, establish the rationale for integration and articulate the schemas in readiness for integration. This entails a high degree of collaboration between data analysts. However, most of the information resulting from such collaboration is often lost because of the lack of mechanisms to effectively capture and document

design discussions. This problem can be attributed to the fact that such information is necessarily semi-structured. The Issue-Based Information System (IBIS) proposed by [Rittel and Kunz 1970] provides a framework within which such collaboration can be supported, and the generated information captured and documented. Hypermedia has also been recognised as a suitable environment for augmenting the IBIS framework.

In this thesis we discuss how the author has investigated the applicability of hypermedia to schema modelling. We also address how the author has extended the investigation of schema modelling to a consideration of collaborative schema integration using an adaptation of the IBIS scheme.

1. Introduction.

This chapter summarises the body of the thesis. An overview of the project is given and a detailed discussion of the research domain and context of the research is provided in section 2. The discussion of the research domain highlights the complexities of database design. The research context discusses related work on the subject, distinguishing it from the author's own work. Section 3 summarises the aims of the research. It overviews how hypermedia and the media facilities it offers may augment database design work. The work done is summarised in section 4, as are the major features of the prototype. Section 5 gives an organisation of the rest of the thesis and section 6 concludes the chapter.

2. Domain and Context of the Research.

This thesis discusses an investigation into the applicability of hypermedia technology to schema modelling and collaborative schema integration. The focus of the work has been on the design of databases for standard information systems. The thesis of the work is that:

a. **Database design can generally be cast as an irregularly structured process.** This irregularity is attributable to the fact that achieving a complete and accurate design is virtually impossible at the first attempt. Further, humans are unable to simultaneously handle vast amounts of information. These factors introduce incompleteness and necessitate iteration in design work.

The major causes of incompleteness are:

i. the format, ordering and number of attributes vary from concept to concept.

ii. users are initially unable to express all the 'desired' data for an information space.

iii. designers are initially interested in skeletal designs for an information space that represent only the major entities and their identifying attributes.

iv. the sheer scale of the task frequently requires incremental design.

Consequently, iterations are necessary to ensure that the design is complete and accurate. Another factor giving rise to iteration is that there are usually many ways of representing the same data semantics. The realisation of a more expressive representation may manifest iteration. Each iteration improves the initial design, providing more detail or modifying it as supporting data becomes available or is clarified.

b. **Schema modelling produces design artifacts that are irregularly linked.** These artifacts and their linkages need to be documented and managed. In documenting schema components, data modellers develop and synthesise designs and encode them in documents of varying format, media (text and graphic), size and granularity. The volume of documents and their inter-linkages compounds the already complex and error-prone database design process. An effective system for the management of such documents is clearly needed.

c. **The logical relationships between schema components (e.g. entities) often necessitate non-linear reading of the design documents.** In particular, the semantically meaningful reading patterns utilise logical chains which help in understanding the data's abstractions, access paths and usage contexts. Clearly, an effective system reflecting the logical authorship and readership of such documents is also needed.

d. **Producing a complete database design for large organisations is a complex process.** By applying a 'divide-and-conquer' approach, this complexity can be reduced. This approach has been widely used in conventional information systems analysis by applying functional decomposition whereby the functional areas of an organisation are identified. This process engineering is easier because organisations tend to have specialised sections, departments etc. each of which contributes towards the total functionality of the organisation. Consequently some functional areas may share common data. A design based on functional areas may therefore duplicate some data semantics. The resultant redundancy leads to problems of consistency and accuracy on the part of the resultant database and must therefore be reduced.

Schema integration is a process during which individual schemas are integrated with the aim of eliminating redundancy. The identification and elimination of redundancy requires co-operation, understanding and agreement among designers and users. Schema integration is based on detailed knowledge of the information areas. This knowledge is often missing in the schemas as they are based on data models, which by definition lack semantic detail. Also, the encoding and acquisition of knowledge normally depends on the perspective of the viewer. These factors inevitably introduce a collaborative dimension to the design process because of its reliance on human interpretation.

e. **Hypermedia is a technology suited to the representation and management of irregularity**. It is a form of information representation supporting irregularly-linked pieces of information of various media. This is achieved by means of structuring and organising the information into chunks and linking them according to some semantic relationship(s).

In summary, the overall thesis is that HyperCard, as a representative of hypermedia technology, lends itself to:

a. supporting the irregular nature of database design tasks.
b. managing the complexity of database design documents.
c. enhancing the collaborative nature of database design work.

The work discussed in this thesis is related to several other research efforts. In terms of schema modelling, HyperCASE [Cybulski and Reed 1992] and Kambanis' work in the TDE (Taxis Development Environment) [Kambanis 1990] are cases in point. For schema integration INCOD [Atzeni et al 1982], IMT [Lundberg 1982], DDEW [Reiner et al 1984], MUVIS [Hayne and Ram 1990], the Cyc-Carnot project [Collet et al 1991] and the tool developed by [Sheth et al 1988] are cases in point. The major distinguishing features of the author's work are that:

a. it supports and provides a framework for collaboration and deliberation, as well as documenting the deliberations.
b. integrations are based on semantic consensus, thus it is fundamentally not a knowledge-based system. It therefore offers limited automation, a feature which positively enhances collaboration.

3. Aims of the research.

The key aim of the research is to investigate the applicability of hypermedia to schema modelling, collaborative schema integration and the associated documentation of database design. Schema modelling and schema integration are the major stages of conceptual modelling, a database design phase in which a data model representing some universe of discourse is produced.

It has long been recognised that hypermedia technology offers a suitable storage and retrieval model for CAISE (Computer Aided Information Systems Engineering) and CAD (Computer Aided Design) [Delisle and Schwartz 1986, Bigelow 1988]. A database is central to modern-day information systems, and its implementation requires considerable design work [Leonard 1992]. Conceptual modelling starts with schema modelling, a stage in which a model that best approximates an information space is produced. Schema modelling generally involves identifying self-contained entities whose data is to be represented in a database along with their relationships. Schema modelling can thus exploit the modularity offered by hypermedia to represent these entities, and indeed the relationships. It seems that representing these self-contained entities in hypermedia nodes not only offers an environment for representing the model, but also for populating the node with additional documentation which would otherwise be difficult. In this light, it is also reasonable to consider hypermedia as a medium for naturally supporting the database design process: design components are identified (entities in our case), (iteratively) designed, represented and linked through relationships.

Besides application to irregularly structured documents, hypermedia has also been widely applied to supporting collaborative work, particularly issue-based environments [Sihto 1989, Streitz et al 1991, Begeman and Conklin 1988]. Collaboration is necessarily social in nature because it involves a number of people working together. Hypermedia has also been applied to CASE (Computer Aided Software Engineering) and CAD (Computer Aided Design) [Delisle and Schwartz 1986, Hayne and Ram 1990, Bigelow 1988]. CASE and CAD are necessarily technical in nature because they tend to follow specific development methods. These applications are further indications that hypermedia may be suitable for collaborative schema integration as schema integration is both technical and human-intensive.

Text and diagrams are the predominant media of schema modelling documents. As far as media is concerned, database design exploits only a subset of the media facilities offered by hypermedia (text, graphics, audio and video).

Thus we utilise only text and graphics in our prototype for the following fundamental reasons:

a. the storage overhead for audio and video is excessive.
b. the technology to support video is expensive. In addition to a video camera and a video recorder, analog/digital interfaces would be needed since computers are digital equipment while video equipment is analog.
c. it is unlikely that data analysts would want to spend their time in recording audio/video data as such work does not visibly contribute to producing a schema.
d. data analysts work in abstraction, and often with things that are not tangible (e.g. accounts, the age of a person) hence may not be recorded visually.

The overall aims of the research are to investigate the suitability of hypermedia technology to:

a. Database design documentation: we focus on the node and linking mechanisms of hypermedia and investigate them with respect to the representation of schema components.
b. Supporting the database design process itself: we focus on the node and linking mechanisms of hypermedia and investigate them with respect to supporting the semi-structured nature of the database design process itself (viz a viz iteration, incompleteness).
c. Collaborative schema integration: we focus on supporting the negotiation aspects of schema integration and exploiting the node and linking mechanisms of hypermedia to document the process and products negotiation.

4. Work Done.

The research method employed has been standard experimental computer science. That is, the construction of a working prototype that demonstrates many of the salient features of a suitable infrastructure for collaborative conceptual modelling. This research has attempted to merge database design activities and hypermedia facilities by developing an active CAISE (Computer Aided Information Systems Engineering) tool for schema modelling, collaborative schema integration and documentation. The demonstrator currently runs on Macintosh hardware using Apple's HyperCard 2.0 and a suite of processing modules developed in Pascal.

The tool utilises an extended E-R (entity-relationship) model and offers the following facilities:

a. a data dictionary which documents the entities, relationships (including the abstraction mechanisms of generalisation and aggregation) and their attributes.
b. an extended E-R diagramming facility for producing diagrams corresponding to the dictionary entries. Diagrams can be segmented to represent large models by utilising an entity clustering algorithm based on the relationships between the entities.
c. an F-D (functional dependency) diagramming facility for producing intra-entity or intra-relationship attribute dependency diagrams. This facility is purely an aid to document the reasons for the selection of keys within the entities or relationships of the extended E-R model.
d. a collaborative schema integration facility designed to support the technical and collaborative aspects of schema integration. The collaborative schema integration facility supports binary integration. A participant in the integration process makes a clear formal proposal for an integration between two selected concepts and gives an explanation. Other participants respond to the proposal, annotating it further with explained agreements/disagreements, questions, comments and data scenarios. This deliberation process can be considered to be validation by consensus, a way of gleaning the semantics of integration and a way of capturing the rationale behind the final integration decision.

5. Organisation of Rest of Text.

The research aims, domain and context have been laid out in this chapter. The rest of the chapters are devoted to hypermedia, schema modelling and schema integration. The technicalities of schema modelling are discussed first. Hypermedia is then discussed and characterised, and a summary case is made for employing this promising technology to collaborative database design. Collaboration is proposed for supporting the schema integration work of integration teams. The IBIS (Issue Based Information System) scheme is introduced and proposed as a framework for collaborative design work. The overlap between database design work and hypermedia culminates in the proposal and development of an integrated CAISE tool incorporating schema modelling, collaborative schema integration and documentation. The tool has two subsystems, the schema modelling system (HSMS- Hypermedia Schema

Modelling System) and the collaborative schema integration tool (SISIBIS-Schema Integration System based on IBIS).

In Chapter 2, the phases of database design are addressed in greater detail. Data modelling is discussed in terms of conceptual, logical and physical modelling (as well as a number of knowledge representation techniques). Focus is shifted to conceptual modelling with detailed discussion of schema modelling and schema integration. The key issues to a successful database design product are also addressed, namely: representation, documentation, communication, understandability, management and collaboration. The chapter concludes by discussing features of hypermedia that render themselves suitable for conceptual modelling work.

Chapter 3 provides an overview of hypermedia technology, particularly its mechanisms and facilities. It discusses the historical background of this technology. The hypermedia environment for the project, Apple's HyperCard is outlined, explaining the features that led to its selection as well as its shortcomings as a hypermedia authoring system. Finally the chapter concludes by proposing the suitability of hypermedia for supporting schema modelling, collaborative schema integration and database design documentation.

Chapter 4 discusses the schema modelling subsystem of the prototype, HSMS (Hypermedia Schema Modelling System). We provide the architecture of HSMS. It focuses on the integrated entity/relationship dictionary facility around which the functionality of the whole tool is centred. The E-R and F-D diagramming facilities are addressed in detail. The chapter also discusses the linking, querying, integrity management and browsing facilities available in HSMS.

Chapter 5 focuses on the process of schema integration. It discusses the need for schema integration and the problems associated with the process. It provides various methodological approaches to schema integration. It discusses in greater detail the major phases of schema integration: preintegration (the comparative analysis of schemas to identify semantic equivalences and semantic conflicts) and merging (the process of implementing the integration among concepts). Finally, it concludes with the assertion that schema integration is an ill-structured process, needing novel ways of supporting it and recording its findings.

Chapter 6 argues for collaborative schema integration. The problems leading to representational ambiguities and redundancies in databases, thus necessitating schema integration, are given. Shortcomings of data models in general are highlighted, suggesting a collaboration-based process for gleaning,

validating and resolving ambiguities as well as documenting the rationale for integration decisions. It addresses the nature and requirements of CSCW (Computer Supported Cooperative Work) in schema integration. It states the stages of schema integration that are amenable to collaboration. It argues for the need to marry CAISE and CSCW schema integration environments so as to store and manage integration models, their derivation and rationale for the integration of their concepts.

Chapter 7 discusses a prototype tool for collaborative schema integration. The IBIS scheme as originally proposed [Rittel and Kunz 1970] is insufficient to support schema integration. The necessary adaptation of the IBIS scheme to support collaborative schema integration is provided. The SISIBIS subsystem is then discussed, highlighting its support for the analytical, technical and social aspects of schema integration. It shows how the prototype is used for knowledge elicitation, semantic enrichment, semantic consensus, rationale capture, merging and documentation.

Chapter 8 details an evaluation of the collaborative subsystem of the prototype, SISIBIS. Both a formal and an informal evaluation are discussed. The formal evaluation was mainly targeted at assessing the utility of the collaborative aspects of the prototype, while the informal evaluation encompassed both utility and usability aspects. Further, the chapter outlines certain problems associated with evaluating the prototype, and concludes by summarising the findings of the evaluation.

Chapter 9 summarises the work. It discusses the limitations of the current work and provides suggestions for further directions of research. Needs for, and shortcomings of, other associated technologies (such as networking, file sharing and hypermedia systems performance) are also addressed.

6. Conclusion.

It is proposed that a contribution to knowledge has been made by the research in three areas where hypermedia appears to render itself to the database design process, namely:

a. process and documentation support: hypermedia appears a suitable environment for supporting the database design process. Its support for irregularity may be useful in supporting incompleteness, iteration and contextual interpretation. Hypermedia also appears a suitable environment for documenting database design information. Its ability to represent

various forms of data of varying granurality seems invaluable. However, hypermedia, as represented by HyperCard, has some general limitations and these are discussed in the body of the text.

b. Collaborative schema integration: the inherent collaborative nature of schema integration is an area that has not received much attention in previous work. It is maintained that complete automatic integration is unachievable because of the human intensive nature of the schema integration process.

c. Relationship between schema integration and abstraction hierarchies: previous work on schema integration has partly considered the issue of abstraction hierarchies (of aggregation and generalisation) as integration constructs. The current work also addresses abstraction hierarchies embedded within input schemas by discussing the implications of integration on existing abstraction hierarchies.

Further areas of research are suggested as arising from this work. The abstraction hierarchies of aggregation and generalisation, and indeed the object-oriented paradigm offer even greater challenges to schema integration. Particularly issues relating to multiple inheritance, polymorphism and integration of behavioural semantics need to be addressed. Versioning is a powerful mechanisms for collaborative work. For schema integration it may be desirable to have a number of versions of the integrated concepts or schemas and then select one based on some quality attributes. The management of such versions is challenging and is an area that needs further research. The issue of versioning naturally extends to or encompasses the problem of design maintenance. We believe both these areas benefit from the node-link mechanisms of hypermedia.

Chapter 2

Database Design and Documentation.

Summary.

This chapter discusses the process of database design with respect to two aspects: the process itself and the documentation of its products. It begins with an introductory section in which we define some terms associated with this process. The discussion introduces database design as a major phase of the information systems development process. Section 2 gives a number of development models. It also discusses database design in the context of these models. Section 3 discusses CAISE (Computer Aided Information Systems Engineering) tool support for systems development in general, with a bias towards database design. The details of conceptual modelling are the subject of section 4. First, we discuss the phases of database design. We then address conceptual and logical modelling. Section 5 discusses conceptual modelling, utilising the Extended-Entity Relationship Model, and CAISE tool support for it. In section 6 we discuss database design documentation. Section 7 concludes the chapter by highlighting the semi-structured nature of conceptual modelling, and suggests a number of suitable requirements for a CAISE environment to support this process.

1. Introduction.

Computerised information systems widely use database systems for the management of large collections of formatted corporate data [Ceri and Pelagatti 1984, Date 1990]. A database system is composed of a database (data store) and a database management system (DBMS) for managing the data.

1.1. What is a Database?.

A database can be defined as a collection of *sharable, non-redundant* data in a *predefined format*. Thus to be a database, a collection of data has to satisfy the following criteria:

a. sharable: data within the system may be used by more than one user with possibly different views of the same data.
b. non-redundant: data is not 'intentionally' duplicated.
c. pre-defined format: data is stored in some carefully structured way, and is not placed in the database haphazardly.

1.2. Database Management System (DBMS).

A database system manages its data through a set of facilities provided by a piece of software called the DBMS (database management system). The facilities include:

a. Kernel (essential) facilities for the proper functioning of the DBMS: definition and manipulation of logical structures, physical storage structures, access and integrity controls, storage management and recovery. Distinguishing between logical and physical structures offers data independence, i.e., buffering the static association between individual application programs and data.
b. Toolkit (auxiliary) facilities such as application programming, data conversion and communication.

1.3. Database design.

Decisions about representing a 'real world' domain by a database system (satisfying the above criteria) are made during a process known as *database design*. A vital stage in database design involves organising data into a *logical model* (an implementation-independent description of the data) which is represented using a *data model* (an abstract model for data). The purpose of a data model is to allow a data analyst to define or describe the semantic properties of a database unambiguously, and to allow the DBMS (which is implemented on the basis of the data model) to process and manipulate the database. A data model consists of structures, operators and constraints defined upon data [Tsichritzis and Lochovsky 1982]. The successful implementation of a database depends on the ability of the logical model to represent reality 'accurately'. A logical model is the culmination of a design process charged with determining the required content of a database. The data analyst extracts such knowledge from the users, who are familiar with the data space but often not conversant with logical data models. Therefore the data analyst has to represent the domain using an abstract model (the *conceptual model*) that is communicable and understandable to the users, yet easily transformed into a logical model supported by the DBMS. Conceptual modelling comprises two stages [Vossen 1990, Navathe et al 1986]:

i. schema modelling which transforms user requirements into individual user schemas, and
ii. schema integration which combines these individual schemas into a single

integrated global schema (the conceptual model).

a. Data Models.

Today four data models are widely recognised: hierarchical, network, relational and the object-oriented models. The first three are however well established i.e. hierarchical, network and relational. These are often referred to as the classic data models [Beynon-Davies 1993] and are illustrated in Figure 2.1. In terms of commercial use, DBMSs have progressed through serial, indexed, hierarchical, network and relational systems. Similarly, the capability and performance of a DBMS depends on the data model it supports [Brodie 1984]. For instance, network models tend to be faster than relational models, but they are less flexible.

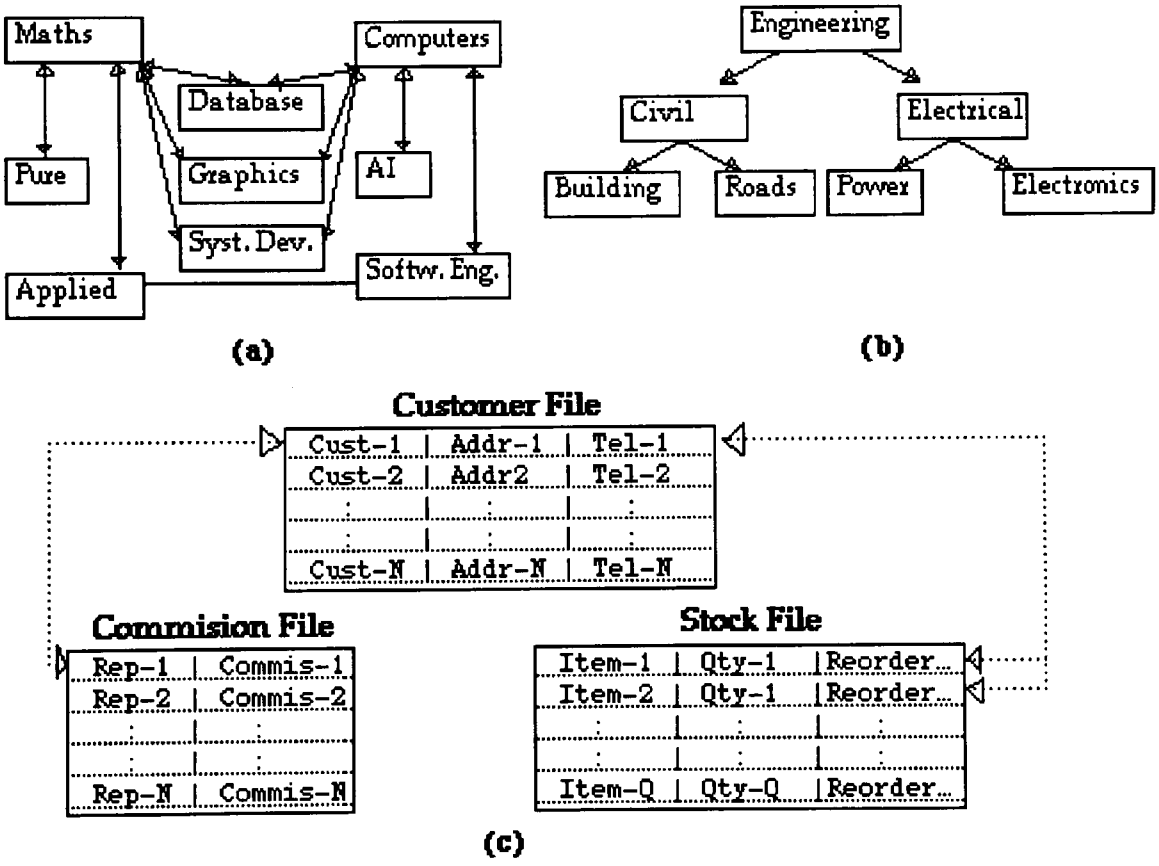

Figure 2.1: Classic Data Models: (a) Network, (b) Hierarchical, (c) Relational.

The hierarchical model organises sets of data entities in strict tree-structured hierarchies i.e. an entity has at most one parent and zero or more children. The link between such entities is thus unidirectional. The hierarchical DBMS (HDBMS) recognises hierarchical models and offers

facilities for defining, managing and accessing entity nodes according to a parent/child relationship.

The network data model is a generalised extension of the hierarchical model which allows an entity to have zero or more parents and children. This means that any entity can be directly related to any other entity through bidirectional links and that entities may share parents and children. In recognising a network model, the network DBMS (NDBMS) offers facilities to define, access and manage data represented according to this model as well as any bidirectional links between them. The hierarchical model is therefore a subset of the network model, and for data exhibiting repeating groups of hierarchical tree-structures a NDBMS can delegate some chores to a HDBMS subsystems.

The relational model exhibits a tabular structure known as a relation and the named columns are termed attributes and the rows are called tuples [Codd 1970]. Similarly, in recognising the relational model, the relational DBMS (RDBMS) manages the definition of attributes, tuples and relations, and offers access facilities to attribute values. Like the network model, the relational model allows bidirectional linking but no repeating groups and in addition uses powerful relational operators (e.g. projection, join) for dynamically combining data from a number of relations to form other relations.

There are a number of the so-called object-oriented models. OO models allow arbitrary user-defined data types known as classes exhibiting hierarchies, inheritance, encapsulation of procedures (known as methods) operating on the data of the class with the data type itself, and dynamic binding. An object-oriented DBMS recognises the appropriate OO model and manages the definition of objects and their relationships as well as access to object instances.

b. Data modelling.

The design effort for large corporate databases requires teamwork and a modular approach [Bishop 1988, Navathe et al 1986]. Design work itself emerges from interactions between team-members, teams and users. Hence it is subject to the effects of human attributes such as creativity, expression and communication [Grudin 1990, Olson and Olson 1991, Rodden 1991]. In particular, these issues arise during schema integration because of the necessity to share data while retaining individual user perspectives to it. Inevitably, during schema integration some schema components are transformed to match others and such change needs to be negotiated and managed. This is the subject of schema integration and is discussed in chapters 5 and 6.

During any design process, data analysts make use of technological artifacts at their disposal. For database development (and systems development in general) the primary technology is a methodology: an organised set of techniques for conducting development tasks [Avison 1992]. Automated design tools (e.g. diagrammers) often support a methodology, and as such are referred to as CAISE (Computer Aided Information Systems Engineering) tools.

Documentation is an integral part of any design process [Bell and Evans 1989]. Data analysts utilise a notation to express the data domain in various documents that encode models and any supplementary information such as notes and annotations. The benefits of documentation are twofold:

i. communicating schemas to stakeholders (persons interested in the schema such as users) as a basis for critiquing, evaluation etc.
ii. recording design decisions, their rationale, alternatives etc.

To conclude, database design involve both technical and social problems. It employs a methodology (c.f. section 2), a notation and possibly CAISE tools (c.f. section 3) and produces various data models (such as conceptual model- c.f. section 4) that are encoded in documents of various format (c.f. section 5). Data models are central to the design process:

'Data models are central to information systems. ... Data models provide the conceptual basis for thinking about data-intensive applications and they provide a formal basis for tools and techniques used in developing and using information systems' [Brodie 1984].

2. Database Design Methodology.

Considerable research effort has been devoted to developing 'better' database design methodologies [Agosti and Johnson 1984, Elmasri and Navathe 1989]. Categorisation of these methodologies can be based on the degree of abstraction (at each step) and formalisation.

Categorising the approaches in terms of their abstraction gradients results in the top-down, bottom-up or middle-out approaches. These are exemplified by E-R-based schema modelling (section 4), the functional approach [Yao et al 1982] and E-R-based schema integration (section 4 and chapter 5) respectively.

An approach can also be categorised in terms of its degree of formalisation: informal, formal or semi-formal. An informal approach has no syntax or

semantics defined upon its notation (e.g. free text) because the individual tokens of the formalism have no underlying meaning. This obviously has several setbacks: no checking can be performed, ambiguity of natural language, no clear distinction between constructs. The formal approach is based on mathematical theory such as predicate calculus and/or logic [Niemi and Jarvelin 1991]. This approach enables checking but its major setback is the need for knowledge of discrete mathematics. The semi-formal (commonly known as the structured) approach, utilises a notational syntax and/or semantics (thus amenable to some degree of checking) and often a graphic notation that amplifies communication. This is the most popular approach and our discussion assumes this approach henceforth.

The structured approach breaks the database design process into the logical stages shown in Figure 2.2. Requirements analysis gleans the data and processing requirements of the database from its anticipated users. Schema modelling organises individual user requirements into a schema, while schema integration merges these schemas into a conceptual schema. The conceptual model is then transformed into a DBMS-independent logical schema which is in turn transformed into an implementable DBMS/application dependent (physical) schema. Logical and physical design are beyond the scope of this work, the text occasionally refers to them in the stated context without further elaboration.

Parallels for systems development models [Bersoff and Davis 1991] can be made in database design. For instance, the structured model of Figure 2.2 is equivalent to the 'waterfall model' [Royce 1987]. A prerequisite of the waterfall model is that requirements must be clear and complete before any proper design work commences [Bishop 1988, Bersoff and Davis 1991]. Bersoff and Davis however note that the waterfall model suffers when the requirements are not clear or are constantly changing. Although this is true for systems requirements, the same can be said for database design requirements, particularly schema integration. For instance, relationships between the artifacts in the input schemas are not clear and have to be carefully established, as well as the possible restructuring (for alignment) before the actual integration. The reuse, prototyping and automated synthesis models are variants of the waterfall model [Bersoff and Davis 1991]. The reuse model incorporates or uses previously defined components. The prototyping model is often used to test pragmatics and encourage user participation, the prototype may end up as the final product via an evolutionary approach. These models have found their way into database design as exemplified by the work of

[Baskerville 1993] on semantic database prototypes. Baskerville's work extended prototyping into the realm of database modelling by arguing that a prototype can serve as an abstraction for conveying a schema.

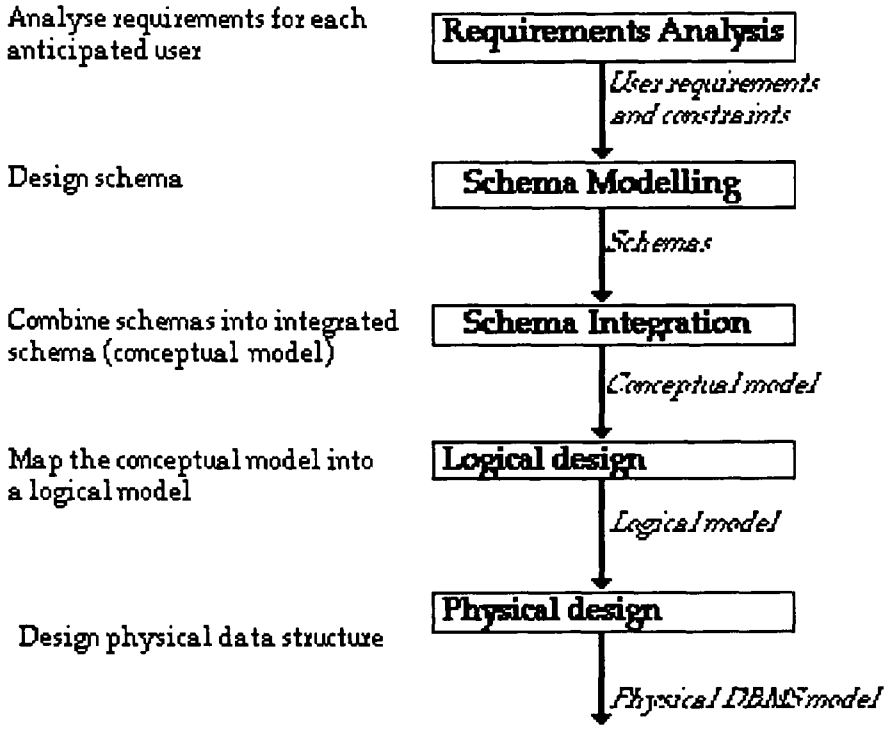

Figure 2.2: Stages of Database Design.

3. CAISE tools: An Overview.

> *'CAISE ... has stimulated the view that information systems development, considered as an information system itself, should be subject to and benefit from the same sorts of automation that characterise everyday information systems. .. the development process is seen as a set of activities operating on objects to produce other objects. The objects manipulated by such activities will frequently be documents, diagrams, file-structures or even programs'* [Beynon-Davies 1993].

CAISE (Computer-Aided Information Systems Engineering) tools are an attempt at computerised support for producing information systems. Editors, assemblers, compilers and debuggers are typical examples of early CAISE tools which were mainly targeted at aiding the programming task. Diagramming tools, data dictionary systems (DDS), prototyping tools, code generators and

integrated project support environments (IPSE) have since emerged to support other elements of the systems development process. CAISE tools are closely related to CASE (Computer-Aided Software Engineering) tools, a distinction can however be made in that, CAISE is a generalisation of CASE.

[Chen et al 1989] includes the following as essential functions of CAISE:

a. Elicitation: the tool must help describe systems at analysis, design and implementation.
b. Analysis: the tool must analyse the consistency and completeness of the data elicited in (a) above and it must notify the data analyst of any detected errors and/or evaluate design alternatives.
c. Information storage: the tool must be able to store relevant data, processes, graphics and rules for error checking, transformations, components etc.

3.1. CAISE and Systems Development.

CAISE tools have been greatly influenced by the emergence of structured methodologies (c.f. section 2). In particular, to gain the full benefit of a structured methodology it is advantageous to utilise computerised tools as evidenced by the 'editing-compiling-linking' sequence in program development. Breaking down the development process into phases with different activities offers opportunities for specifically tailoring a tool to the activities of the phase as well as supporting an appropriate notation. CAISE tools have therefore been developed to support specific phases of the systems development life-cycle. The tools can be categorised in terms of the following characteristics:

a. Life-cycle coverage: CAISE tools vary in their coverage of the development life-cycle. Most tools are either front-end tools (analysis, planning, design) or back-end tools (programming, debugging, testing, implementation).
b. Activity support: CAISE tools may be categorised as administrative or technical depending on the activities they support in their life-cycle scope [Butler 1987, Winsberg 1988]:

   i. Administrative: management of resources (budget, schedules, manpower etc.).
   ii. Technical: support the technical activities of the development process (design, prototyping, code generation, documentation, training, collaboration etc.).

CAISE tools may also differ in the depth of support they provide in their respective areas. The depth of support may include among other issues validation, analysis, critiquing, multi-user and tailorability. The common approach is to provide full support or an aggregation of various support facilities. Another approach involves adapting a methodology by configuring or adapting the CAISE tools (such as altering the symbolic notation). Such tools are referred to as meta-CAISE tools [Hsu et al 1991]. Recent approaches aim at integrating a number of CAISE tools to cover various development stages. Integrating tools from various vendors results in C-CAISE (Component CAISE). On the other hand, integrating tools from the same vendor that are meant to work together results in I-CAISE (Integrated CAISE). IPSE (Integrated Project Support Environments) [McDermid 1985] can be considered as early I-CAISE systems.

Most CAISE tools utilise a dictionary (often referred to as the systems development encyclopaedia) to store the appropriate development information. The dictionary is often viewed as a separate tool. In the database realm, this tool has recently come to prominence with the recent emphasis on repositories. The dictionary is used as a conventional productivity tool supporting the life-cycle of database systems or as a tool for information resource management [Zachman 1987, Jones 1992, Cybulski and Reed 1992]. A data dictionary contains meta-data (data about data) such as operational and control information. For our discussion, the meta-data at least includes the schema information for the intended database and their respective data elements. [Hsu et al 1991] note that three basic elements are necessary for the dictionary system:

a. a logical model to define and represent the meta-data.
b. a physical storage structure derived from this representation.
c. a meta-database management system providing facilities for effective implementation and processing of meta-data.

4. Conceptual modelling.

In this section we discuss conceptual modelling with an emphasis on data models. A number of popular data models are overviewed together with their suitability for conceptual modelling.

4.1. Requirements for conceptual modelling.

The following are essential to the expressive power of a data model:

a. Support for abstraction: Abstraction is the simplification mechanism used to hide superfluous details- showing only the important aspects of the enterprise. This allows developers to concentrate on the properties that are essential to the problem domain. The major data abstraction constructs are classification, aggregation, generalisation and association [Brodie 1984, Mylopoulos and Schmidt 1984]:

   i. Classification is the mechanism by which concepts are grouped on the basis of likeness or conformance to some identifiable type. This allows 'real-world' objects to be identified as instances of that type.
   ii. Aggregation constructs group concepts together to form a higher-level construct. This construct is particularly useful in representing concepts that are physically or logically 'part of' another concept. The physical view often pertains to composition (a car is composed of doors, wheels etc.) while the logical view often pertains to properties (name, age and sex are properties of a person).
   iii. Generalisation constructs create concepts that are subtypes (or supertypes) of other concepts. The subtype inherits the properties of the supertype and the subtypes themselves specialise the supertype in a mutually-exclusive or overlapping manner [Elmasri and Navathe 1989]. For instance 'undergraduate' and 'postgraduate' are subtypes of 'student'.
   iv. Association constructs relate concepts on the basis of equal syntactic weight i.e. no concept dominates another. A 'student' taking a 'course' is an association relationship.

b. Notation: a clear notation which supports the data model. The notation must be easy to understand so as to assist the data analyst in communicating and interpreting ideas and models. A notation may be textual, graphic or mathematical (formal). A textual notation uses semi-structured English (e.g. the works of [Chen 1983] and [Eick and Lockemann 1985] are based on natural language processing to produce schemas). A graphic notation uses graphic symbols (e.g. E-R diagrams: c.f. subsection (a) below) and a mathematical notation uses predicate calculus/logic or set/algebraic theory (e.g. Z [Bryant 1989]).

c. Documentation support: document the intentions of the data analyst to maximise readability, understandability and maintainability. The support for good documentation can be achieved by:

i. use of meaningful symbols and object identifiers,
ii. good presentation of the data model to clarify the structure the data.

d. Integrity checking: identification of constraints and implementing the data model in compliance with the constraints.

4.2. Data Models for Conceptual Modelling.

The classic data models have been discussed as being unsuitable for conceptual modelling [Teorey et al 1986]. This has been mainly due to their inadequacy in capturing data semantics in ways that enhance thinking about data [Kent 1991] (i.e. lack of (a) above). The hierarchical model naturally models the semantics of hierarchical organisations but suffers in its ability to model non-hierarchical domains. Its use in non-hierarchical domains imply that to conform to a hierarchy, construction rules may be satisfied only by the introduction of artificial concepts. The relational model [Codd 1970] has been criticised because it's constructs require cumbersome logical transformations between the 'real-world' and relations (tables) [Codd 1979, Elmasri and Navathe 1989, Smith and Smith 1977]. This again is mainly due to the creation of artificial entities to accommodate multivalued functions between entities e.g. the so called 'foreign keys'. Consequently, a number of semantic data models have emerged [Hammer and McLeod 1981, Chen 1976, Smith and Smith 1977] which offer more natural support for the representation and conceptualisation of data domains.

a. Semantic Data models.

Semantic data models provide the data analyst with better facilities for conceptual modelling than the classic models. A semantic data model (like any data model) consists of structures, operators and constraints defined upon data. The structures represent data or activities that operate on the data. These include data items, entities and interactions between them. Operators represent processes that update the data or retrieve data in response to some enquiry. Constraints are restrictions on the states of data that must hold true for all states of the intended database. Structures therefore form the bed-rock of semantic modelling.

Many semantic data models incorporate the concepts of aggregation, classification, generalisation and association [Brodie 1984] (cf. 4.1) in various combinations. These include the entity-relationship model (ERM) [Chen 1976] and its extensions (EERM) [Teorey et al 1986, Hammer and McLeod 1981], the

functional model [Shipman 1981], and the recently emerged object-oriented models (OOM) [Shlaer and Mellor 1988, Stefik and Bobrow 1986]. The next sub-sections describe these models.

The Functional Model.

The functional model [Shipman 1981, Yao et al 1982] recognises classifications and the functional relationships between the identified concepts. It consists of functions defined on data sets, where a function is defined as a named set-valued map from one set (the domain or argument) to another (the range). Figure 2.3 depicts a functional model. Because it is based on the mathematical notion of a function (i.e. mapping into only one range-value) this model fails to model non-functional relationships naturally (such as that between a 'vehicle' and a 'driver' in the figure). The mathematical theory of functions is not so natural to both users and data analysts, hence the functional model is unsuitable for conceptual modelling. Functional theory has however been widely used for normalisation [Kent 1983]. Normalisation is a technique that reduces anomalies (side effects) associated with operations (insert, delete, update) on data groups exhibiting some characteristics.

A model that consists of both functional and non-functional maps captures the determinancy (dependency) between the involved data items [Desai 1990]. The domain data item is termed the determinant while the range data item is termed the dependent. In Figure 2.3 'driveLesson' functionally determines 'driver'. Such a diagram is said to be a dependency (determinancy) diagram.

Dependency analysis can be considered a bottom-up approach to conceptual modelling. Having identified a data item, the data analyst, through dependencies, can gradually extend the boundary of the concept the data item represents. Each boundary can then be considered to be a first-cut representation of the concept (entity) (that is uniquely identified by the data item on which the boundary is based). Hence dependency theory can be used to complement top-down approaches such as the E-R approach discussed below.

ii. The Entity-Relationship Model (ERM).

Of all approaches, the ERM [Chen 1976] is the most widely accepted for conceptual data modelling. The ERM recognises classification and association only: it views an enterprise as consisting of entities that are linked by association relationships with constraints defined upon them. Entities and relationships are described by attributes. Two types of constraints are represented: cardinality and optionality. They both provide constraints about the participation of member entities in relationships. The cardinality shows

the number of unique instances of one entity associated with the other while optionality shows the existence dependency between member entities. Relationships are usually binary (involving only two entities) but n-ary relationships can also be modelled. In Figure 2.4 'undergraduate' and 'course' are entities linked by the binary relationship 'takes'.

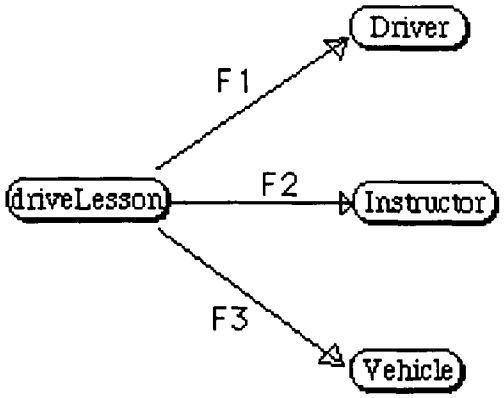

Figure 2.3: An Illustration of the Functional Model.

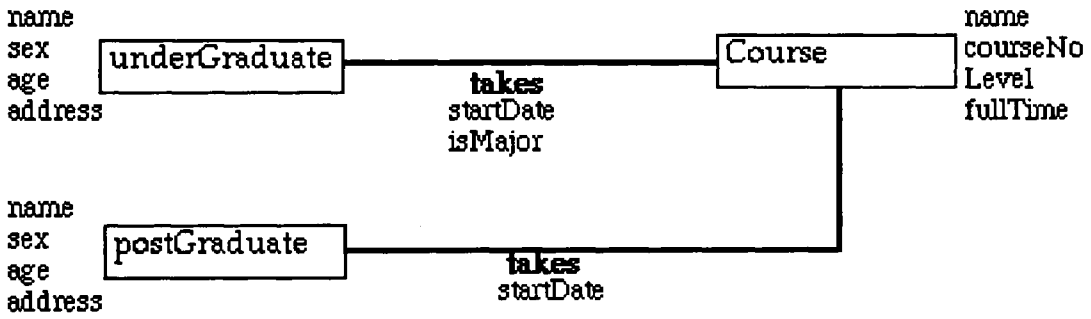

Figure 2.4: An ER Model Depicting the Link Between Students and Courses.

However, the ERM has been justifiably criticised for its lack of support to fully capture the semantics of data and the intentions of the data analyst (e.g. generalisation/specialisation). For instance, in the figure it fails to capture the commonalities between the entities 'undergraduate' and 'postgraduate' and the two 'takes' relationships. A number of extensions to the ERM incorporating aggregation and various forms of generalisation have been developed. These are generally referred to as the Extended E-R models (EERM) and include the SDM (Semantic Database Model) [Hammer and McLeod 1981], LRDM [Teorey et al 1986] and ECER [Czejdo et al 1990].

iii. The Object-Oriented Models (OOMs).

The OOMs are based on the concept of an object: an encapsulation of data and its behavioural semantics (methods) that is capable of receiving messages that invoke its behaviour. Besides encapsulation, OO modelling offers information hiding through inheritance (via generalisation). From a data modelling perspective, object-orientation provides a modelling framework offering modularity, reuse, fewer constructs (in the form of data and methods only) and information hiding. The incorporation of methods allows us to model constraints and transactions defined upon the data. OO models can be considered as yet another improvement on the E-R model (EER in this case) by the incorporation of methods, replacing the term entity with 'class' and attribute with 'class variables'. Indeed, suggestions have been made of naturally extending ER modelling to OO modelling [Beynon-Davies 1992(a)] as shown in Figure 2.5. The OMT (Object Modelling Technique) [Loomis et al 1987, Blaha et al 1988] is one such suggestion. However, there is as yet no agreed OOM as the OO paradigm itself is not firmly established.

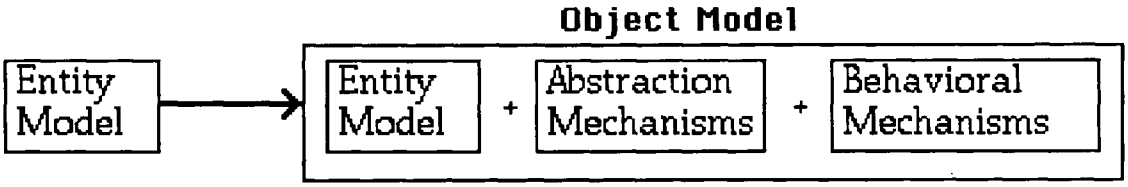

Figure 2.5: Transition from ERM to OOM [Beynon-Davies 1992(a)].

Even the OOM has been cited as unsuitable for conceptual modelling. [Barclay and Kennedy 1991] in a paper entitled 'Regaining the Conceptual Level in Object-Oriented Modelling' argue that the OOM results in loss of conceptual level and question the incorporation of constraints into a single framework. The OO paradigm has not established itself as much as the ERM despite its promising attractions.

5. Conceptual Modelling using E-R approach: CAISE tool support.

The ERM is the most popular approach to conceptual modelling and the work discussed in this thesis assumes the EERM. Figure 2.6 shows two schemas, one relating an undergraduate student to a course (a) and the other relating a postgraduate student to a course (b) and the two integrated back into one schema (c). What the integrated schema does not show is the rationale for integration, which must be included as part of the design documentation. The need for CAISE tool support for such design work has been discussed [Fogel

1992, Siau et al 1992, Hayne and Ram 1990]. The production of diagrams may be aided by diagramming tools [Martin and McLure 1985] while schema integration may be supported by decision support tools such as prototypes [Baskerville 1993], expert systems [Hayne and Ram 1990] and group decision support systems such as argumentation systems [McCall 1988, Rittel and Kunz 1970]. Above all, the work and all associated information has to be documented in a format that allows flexible access and representation of all the relevant information. These are only a few cases of CAISE application to this complex and ill-structured process. Additional tools may offer analysis (semantic, syntactic, natural language).

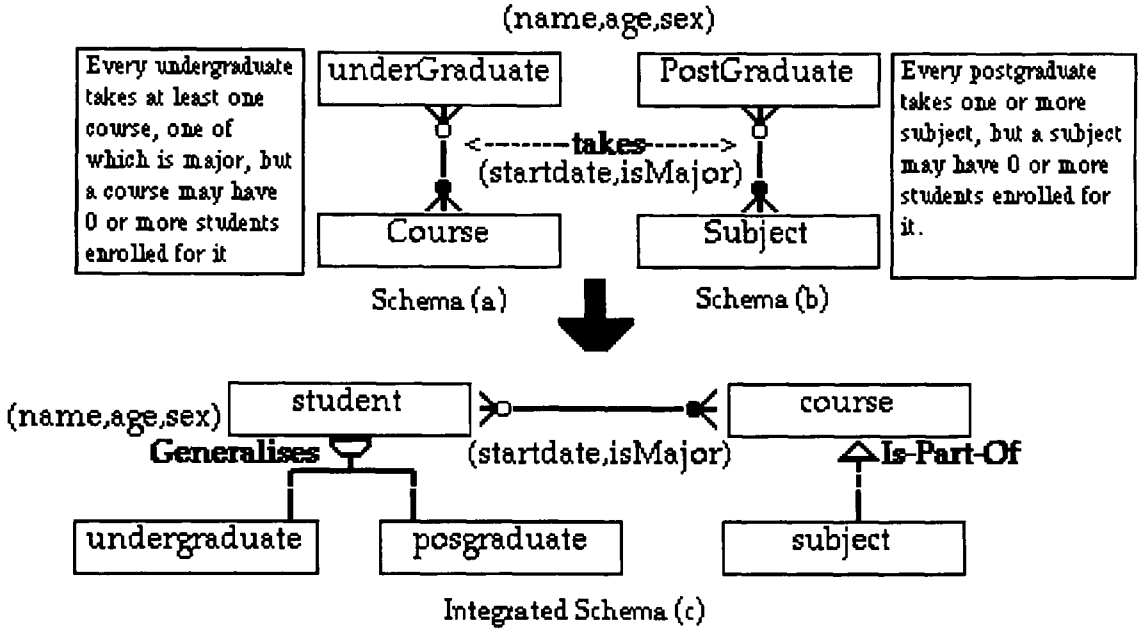

Figure 2.6: Sample Schemas and their Integration.

There are a number of competing notations for EER diagramming, and the most popular ones are due to Martin and Bachman and the discussion will assume Martin's (the crow-foot) notation. Figure 2.7 shows this notation in addition to that used in this work for the abstraction mechanisms of aggregation and generalisation. Entities are represented by rectangles, relationships by arcs and relationship constraints by icons attached to the member entities of the relationship. A cardinality of one is represented by a single-line icon, otherwise it is represented by a crow's foot. Attached to the cardinality icon is the optionality icon: mandatory participation is indicated by a plain circle, otherwise it is represented by a shaded circle.

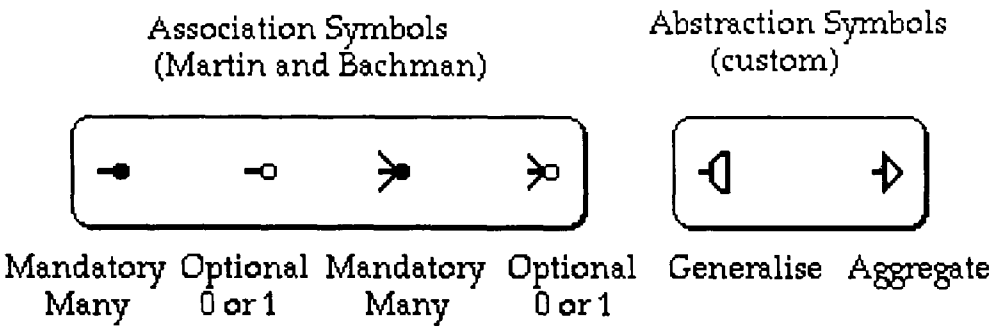

Figure 2.7: HEERD's EERM Diagramming Notation.

6. Design Documentation.

Documentation is an integral part of any design process [Bell and Evans 1989]. Data analysts encode design models and any supplementary information (such as notes and annotations) in documents of various format. Central to the documentation process is a data dictionary: a repository in which the models and design information are stored. The main purposes of documentation are recording of design models (for communication), design decisions and their rationale.

a. Communication.

Design, a process that creates objects that satisfy a set of requirements, can be considered to include three major aspects: analysis, synthesis and evaluation [Olerup 1991]. For database design, the underlying requirement is that the resultant database reflects the 'real-world' state of the target domain. The data analyst therefore gleans domain information from the users (analysis), logically maps the information into a data model construct (synthesis) that is known to satisfy given architectural requirements (e.g. entity, relationship, attribute) and encodes it as such. To ensure the document correctly represents the intentions of the users, the model is communicated back to the user for approval (evaluation). Thus during this analysis-synthesis-evaluation cycle, documents play a major role in achieving a communicable correct representation. The 'producer-consumer' relationship between the design phase and subsequent phases of the development process entail that at the end of each major design phase a deliverable (a project milestone) is produced. Communication is often aided by a notation that expresses data semantics. For instance, an ERM is often represented by an ER diagram.

b. Recording.

Design recording serves as a storage mechanisms for design models, decisions and rationale. The sheer volume of data and their semantics forbid designers to hold all models in their heads, otherwise they would not remember all the information and how they were synthesised. Storing the information relieves the data analyst of this burden. However, they still need to retrieve the information: obviously the knowledge requirements for this are far less and can be supplemented by organising and searching techniques. Maintenance problems necessitate recording of design models so that the models may be reused and do not need to be recreated each time.

The above discussions ((a) and (b) above) are equally sound for conceptual modelling. Schemas need to be recorded so they can be communicated for schema modelling and schema integration, and also to serve as design knowledge-bases for subsequent work and integration rationale. The abstraction employed in conceptual modelling, however, complicates design documentation insofar as communicability is concerned. Abstraction appears in various guises, from linguistic abstractions (naming things), graphic abstractions (symbolisms) and the data-abstracting mechanisms themselves. This system of language and signs compounds model communication and has been addressed by a number of authors [Beynon-Davies 1992(b), Baskerville 1993] thus affecting the utility of the documentation. The abstraction mechanisms (classification, aggregation, generalisation and association) entail some degree of modularity which similarly reflects on the documentation. The sheer number of design artifacts complicates documentation authoring, representation and communication. [Walker 1991] put it aptly:

> *' ... bigger documents differ in nature from smaller ones. It is significantly more difficult to write a report than a memo and more difficult yet to write a large manual than a report'.*

Examining a database design document and its use exhibits interesting characteristics:

i. it is not single-media, but rather multi-media by virtue of the presence of text and diagrams.
ii. modularity is explicit, necessitating modular documentation. As in

programming, internal (along with the individual document components) and external (supplementary to the model) documentation are common. Excessive internal documentation does clutter a document and hamper its communicability and that of what it is intended to document. Implicit in the documents are inter-linkages between the components based on use, abstraction or semantics.

iii. non-sequential navigation through the multi-media document is used in reading, discussing and referencing components within the model. The links that define the navigational paths are mainly predefined and explicit (e.g. relationships, references), but often they are implicit.

New ways of managing such documents and their usage are therefore needed.

7. Conclusion.

This chapter provided an overview of database design and documentation. We discussed database systems as consisting of a DBMS and a database, and noted that a data model is central to both. Approaches to database design were also outlined and we showed how a systematic (e.g. structured) approach to database design and the use of data models make the database design process suitable for CAISE tool support. The functional model, the EER model and the OO model were discussed. Despite a systematic approach, database design was shown to be semi-structured, iterative, often in a state of incompleteness and requiring collaboration. The need for an environment to support such processes was identified.

The implications of modularity and abstraction in database design and the associated documentation were introduced: new ways of managing the documents and their usage are needed. Specifically, a multi-media environment supporting hyper-dimensional navigation is required.

The next chapter discusses a computerised environment that may satisfy the identified requirements: hypermedia. Hypermedia is a system of information management based on the principle of connecting multi-media information (text, graphic, video, audio) by associative links that can be traced. Hypermedia seems to render itself to database design work: its products are mainly in textual and graphic form, it can support the irregular process, abstraction (through pieces of information: modularity), support a data model (by encoding it: entities, attributes) and offer many presentational options (text, graphics).

Chapter 2

Chapter 3

An Overview of Hypermedia Systems.

Summary.

This chapter gives an overview of hypermedia systems. It begins with an introductory section which outlines the principles of hypermedia. Section 2 discusses the historical background of this technology, as well as sample representative systems and their domains of application. The architecture, mechanisms and facilities of hypermedia are addressed in section 3. Section 4 describes the hypermedia environment for this project, HyperCard. The rationale for the choice of HyperCard as the hypermedia environment is given, as well as a description of its shortcomings. Section 5 concludes by proposing a hypermedia environment for database design, documentation and collaborative schema integration. Section 6 concludes the chapter.

1. Introduction.

In a paper written in 1945 [Bush 1945] laid the foundations for modern day hypermedia systems in his vision of the 'memex':

> *'a device in which an individual stores his books, records, and communications, and which is mechanised so that it may be consulted with exceeding speed and flexibility'.*

[Nelson 1965] coined the termed 'hypertext' for the technology underlying such a device and defined it as 'non-linear reading or writing'. In this section, we define the general features of hypermedia and then distinguish the term hypermedia from the terms hypertext and multimedia.

Hypermedia is a form of information representation and management based on the principle of connecting a number of nodes with links. The nodes may contain various media such as text, graphics, audio and video. This makes it an attractive environment to represent, organise and manage information-rich documents.

The strength of hypermedia lies in associative linking, i.e., partitioning information into nodes and linking these nodes together with a series of semantic and/or syntactic relationships. Nodes are generally considered as self-contained chunks of information. The entire set of nodes relevant to a domain is known as a docuverse. The entire set of nodes plus links is known as the hyperdocument. Links may be considered as semantic and/or syntactic constructs. Semantically, links are the mechanisms of forming an associative

store of information. Syntactically, links impose a non-linear logical structure on the otherwise linear physical structure of a document.

A number of people have suggested that hypermedia environments may mimic features of the human brain [Bush 1945, Engelbart 1968]:

> *'It is an enlarged intimate supplement to his memory... The human mind .. operates by association. With one item in grasp, it snaps instantly to the next that is suggested by the association of thoughts, ..'* [Bush 1945].

The relationship between human associative memory and hypermedia systems has led many to propose hypermedia as a means of augmenting human cognitive activity:

a. through its ability to do routine tasks faster e.g. access to information.
b. by enhancing analytical thinking. The ability to assemble nodes in new ways via links enhance the synthesis of ideas or components.
c. through its support for human memory.

The terms hypermedia, multimedia and hypertext are often used interchangeably. However, a general distinction can be made in that while hypertext denotes a set of loosely-connected textual systems, hypermedia and multimedia denote more elaborate systems by offering other media such as graphics, audio and video. The term multimedia is frequently used in a more technological sense to refer to the encapsulation of various media (multi-media) coupled with developments such as CD-ROM etc. Thus a node can be viewed as a multi-media resource. Hypermedia then refers to multimedia with linking, and hypertext is a subset of hypermedia. This text will use the terms hypermedia and hypertext interchangeably as the prototypes described only exploit text and graphics.

2. The History of Hypermedia.

As indicated in section 1, the foundations of hypermedia are accredited to [Bush 1945] in his description of a device he termed the memex (*mem*ory *ex*tender). As scientific advisor to American President Roosevelt, Bush envisaged the memex as a device through which the body of scientific knowledge could be stored and accessed speedily and flexibly. The technology of the day (microfilm and photo-electric cells), however, restricted its development. These limitations were soon eliminated with the coming of the

computer age. Computers offered fast text-handling (later graphics, audio and video) and logical access to data storage media which allowed both linear and non-linear access to data. Information could therefore be accessed quickly according to associative access paths defined upon it. Other technical developments soon contributed to the realisation of Bush's vision. The advent of large storage devices capable of storing graphic, audio and video data encouraged incorporation of these media in computerised information systems. The development of high-resolution graphics allowed computer systems to provide different text fonts and graphic displays (pictures, animation and windows). [Nielsen et al 1993] have noted that direct manipulation is fundamental to most current graphical user-interfaces (GUIs) and that *'direct manipulation requires that the objects to be manipulated are made explicit to the user and represented visibly on the screen'*. Information may be visibly displayed on a node by node basis in various ways (scroll bars, icons, symbols, windows etc.). For instance, the development of windowing system software allowed computers to display related information units on the screen at the same time. Consequently, a number of systems exhibiting Bush's vision began to appear.

2.1. Sample Hypermedia Systems.

[Fiderio 1988] has categorised hypermedia systems into on-line browsing systems, literary-exchange systems, problem-resolution systems and general-purpose systems.

a. On-line browsing systems are meant to browse through a hyperdocument. Node and link editing features are rudimentary as the information is predefined and pre-linked. Systems falling in this category include on-line references, on-line manuals or help, CAI (Computer-Aided Instruction) systems and guided-tours.
b. Literary-exchange systems [Yankelovich et al 1985] support on line-libraries offering writing, storage and critiquing of literature documents. Xanadu [Nelson 1987] and Textnet [Trigg and Weiser 1986] are typical examples. Both systems offer collaboration while Xanadu also supports versioning.
c. Problem-resolution systems support problem definition and analysis through structured node-linking mechanisms. Often these tools offer view filtering (suppression of undesirable detail) to tailor information to the reader's needs. Augment [Engelbart 1968] is the earliest of such systems. It stores information in a hierarchical structure allowing non-hierarchical

branching. Augment was innovative because of its inclusion of structured editing, mouse-controlled cursor manipulation, distribution and view filtering. The main vehicle for analysis is structuring the nodes into clear hierarchies supplemented by non-hierarchical branching. The gIBIS [Begeman and Conklin 1988] and PHI systems [McCall 1988, McCall 1991] went further by adding an argumentative dimension to problem analysis in capturing design rationale. These systems utilise a framework supporting argumentation called the IBIS (Issue-Based Information System) [Rittel and Kunz 1970]. Briefly, the IBIS scheme organises information into nodes corresponding to an issue (the problem), a position (solution to problem), an argument (rationale for position) and a resolution (problem solution). These nodes are linked into an associative store for the problem in question by links of the following types (3.2 (c)): 'responds-to', 'supports', 'objects-to', 'generalises', 'specialises', 'question', 'suggested-by', 'replaces' and 'other'. Propositions to extend such systems into the realm of CSCW (Computer Supported Cooperative Work) applications [Streitz et al 1991] have been made, culminating in projects such as TeamWorkStation [Ishii 1991] and that discussed in [Fischer et al 1992].

d. General-purpose systems are customisable to the application domain. NoteCards [Trigg et al 1986, Halasz 1988] is the first of such systems. The development of NoteCards was inspired by the practice of creating 'notes on cards'. NoteCards first introduced the concepts of keyword selection, icons, card-layering and graphic-network displays. Many hypermedia systems with the feel and look of NoteCards later appeared (such as HyperCard [Apple Computer Inc. 1990] (c.f. section 4), GUIDE [Harriman 1987]). NoteCards and its related systems offer a more appealing vision of hypermedia because of their use of text, buttons, graphics and object layers. These offer more elaborate ways of organising, displaying and linking information. However, they tend to demand excessive storage requirements and expertise in their construction.

3. Components of hypermedia systems.

A typical hypermedia system consists of nodes and links as shown in Figure 3.1. Underlying the hypermedia systems is a database engine that provides storage and access mechanisms for nodes and links. The user interface is the top-most layer offering the user views of nodes and the links connecting the nodes.

3.1. Nodes.

The basic working unit of hypermedia systems is the node. This represents at the lowest level a single concept or idea. Nodes are both syntactically and semantically discrete. A hypermedia system provides node editors offering creation and deletion of nodes as well as optionally offering text, graphic, audio and video editing to modify information contained within nodes. Three issues pertain to nodes, namely representation, organisation and presentation.

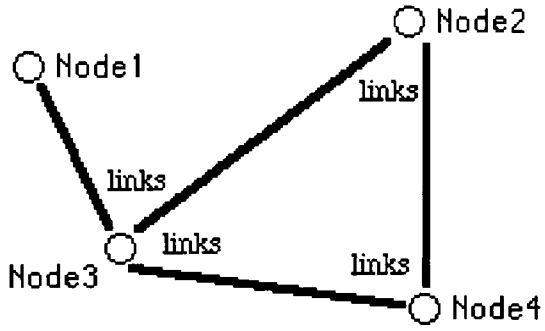

Figure 3.1: Components of Hypermedia.

a. Representation.

Nodes need the ability to represent discrete chunks of information. We may generally distinguish between typed and untyped nodes [Conklin 1987, Begeman and Conklin 1988, McCall 1988]. There are two schools of thought on this issue: one is structural and the other semantic.

The structural school attempts to categorise a node on the basis of its medium: text, graphic etc. An untyped node can hold any media whereas a typed node is restricted to contain only information of a specific type. Typing nodes has the advantage of classifying or aggregating nodes (c.f. contexts (b) below).

The semantic school attempts to categorise a node on the basis of how the information it contains is logically used. For instance, in gIBIS [Begeman and Conklin 1988] a node is classified as an 'issue', 'position', 'argument' or 'resolution' node.

b. Organisation.

The concept of contexts [Delisle and Schwartz 1987] has been proposed as a means to partition and collect nodes and links into sets. This concept has been used to address issues pertaining to design work: collaboration, version management, local and global workspaces as well as configuration

management [Delisle and Schwartz 1987, Bigelow 1988]. A simple use of contexts is as organising themes that group nodes and links on the basis of node types. In this way, a node that is composed of a collection of nodes is formed. Such a node is referred to as a composed node. For instance, Figure 3.3 shows the 'entity context' and 'relationship context' nested in the 'entity-relationship' context as well as the 'entity instance context' (the user-interface) nested in the 'entity context'. Operations such as edit and browse can then be context-specific. For instance, a delete operation would delete a composed node and all its sub-nodes, whereas browsers may be restricted to within the composed node only.

c. Presentation.

How information is presented to readers determines its understandability and communicability. Information that is cluttered is not easy to understand, let alone communicate the knowledge embedded in it for the reader has to sift through the document in search of some knowledge. Hypermedia may offer document presentation facilities, such as annotations, narrations, browsing, zooming, transparencies and process simulation [Christodoulakis et al 1986]. All the above capabilities exploit the node-linking mechanism of hypermedia:

i. Annotations: annotations provide notes, descriptions, explanations and related information on a particular subject. Annotations can be judiciously combined with transparencies (c.f. (v) below) to enrich a presentation. An annotation may be associated with one or more nodes, and in the former case they form the basis for contexts.
ii. Narrations: an association between a voice segment with a particular node that is often switchable.
iii. Browsing: allows the reader to traverse the hyperdocument following the links between nodes. Often browsing exploits the organisational structuring of nodes into subnodes by displaying browsing subnodes on fixed locations of the containing node, thus the browsing method is apparent to the reader.
iv. Zooming: increases or reduces the detail of information that already exists.
v. Transparencies: the ability to overlay nodes may be used for presentation. For instance one could overlay a picture with an annotation to enrich the presentation.
vi. Process simulations: the document author 'programs' the node to simulate some process or behaviour i.e. animation. This may be accomplished by page-turning commands, scrolling, zooming, wiping etc.

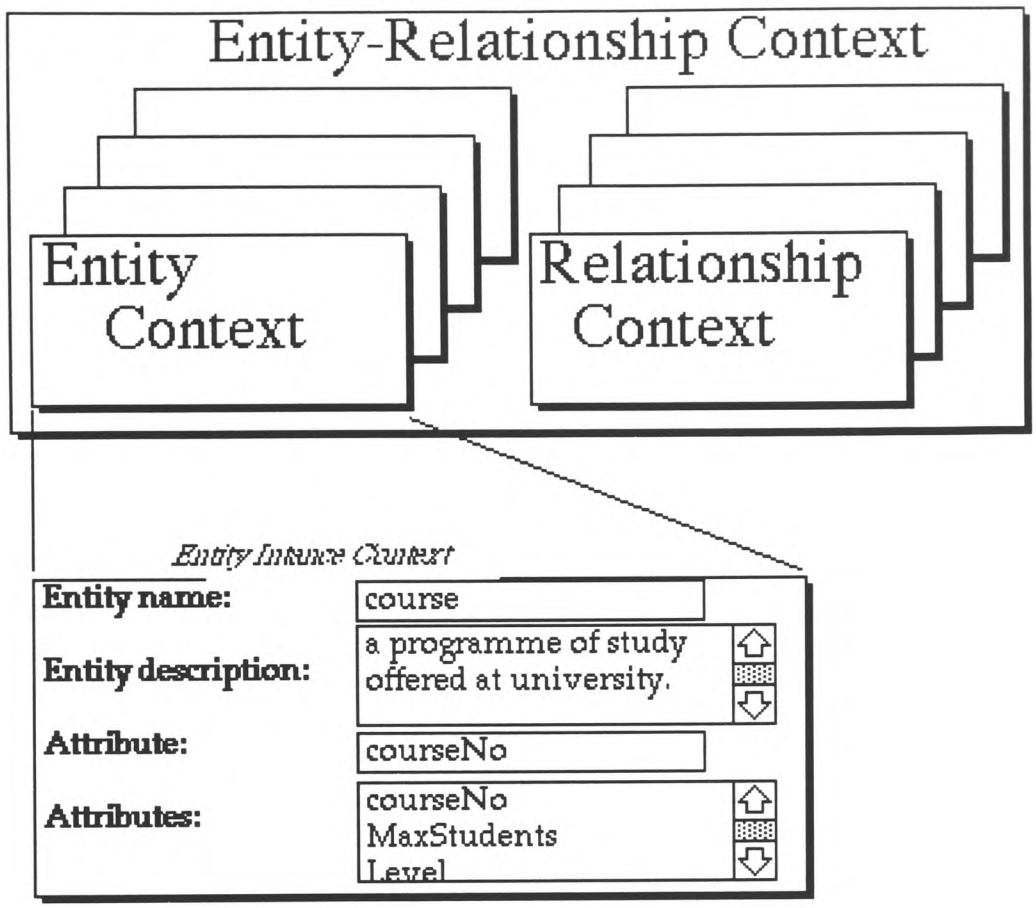

Figure 3.3: An Illustration of Contexts in the ER Domain.

3.2. Links.

Links are used to represent an association between nodes in a document. Links must possess the following properties [Fiderio 1988, Conklin 1987]:

a. representability: links must be represented, physically or logically, in the computer. The linking information may be embedded within its member nodes or a separate node may be designated to represent the link. Link nodes are often used for complex links i.e., besides indexing, the link has additional intrinsic information associated with it (c.f. relationships).
b. relationality: a link must relate at least two nodes, known as the anchors of a link. An anchor is the area of the docuverse to which a link is attached. This is typically a complete node though it can be a distinct part of the information contained in a node. The relationality of an anchor is often defined by its valence, i.e., the number of links emanating from it. A univalent anchor has only one link attached to it, while a multivalent anchor has more than one links attached to it.

c. traceability: the hypermedia environment must be able to directionally trace a link. Unidirectional links can be traversed in one direction only because there is no converse link. Bidirectional links can be traced in either direction. Links may therefore be considered as directional arcs in which case the anchors are respectively referred to as the source and destination of the link.

Normally the hyperdocument author creates the links using a link editor but some systems create links automatically. Automatic (or virtual) linking is helpful in cross-referencing large databases to avoid burdening the author with creating the links manually. Links can normally be classified as structural, user-defined or virtual links as described below. A link editor supports the creation and deletion of links.

3.2.1. Structural links.

These are the links which maintain the underlying order of the hyperdocument. They enforce the mapping between the conventional document and the hypermedia structure and are maintained by the database engine. For instance, structural links automatically attach sub-nodes to composed nodes as well as any data associated with them.

3.2.2. User-defined links.

These are the links which allow the creation of non-sequential paths through the hyperdocument. User-defined links generally fall into four categories: note, replacement, reference and graphic [Conklin 1987]:

a. Note link: A note link can be used to provide annotation. It usually pops-up in a window on the screen and is not transitive, i.e., does not link to additional information.
b. Replacement link: A replacement link is aimed at providing better support for the user by replacing information at the link point with more detailed and elaborate information.
c. Reference link: A reference link is used to identify possible branching-off points in a document, leading to separate information. It can be said to provide global support as opposed to local support for replacement links.
d. Graphic link: a link that is a label or part of a picture. Graphic links give a pictorial view of the document, either globally or locally. Activating a graphic link may produce additional information pertaining to the link. The

addition of animation makes the pictures more informative, hence increasing their value.

3.2.3. Virtual links.

Structural links are generally static links, i.e., links whose anchors are physically embedded within the nodes of the hyperdocument. Such links suffer from a number of limitations. For example, [Halasz 1988] and [Bieber 1992] point out that the responsibility for link generation and maintenance is placed entirely upon the hyperdocument author. Link creation and maintenance problems grow exponentially with the size of the hyperdocument, and typically hyperdocuments tend to be very large. Too many links compound the highly documented problem of 'getting lost in hyperspace' [Dillon et al 1990] for the hyperdocument reader. Further, for some environments adaptive-linking is desirable so that links are tailored to the user's current requirements. [Tomek and Maurer 1992] have proposed the idea of link filtering by user profile. Here a user profile detailing areas of interest, user-groups, preferred media etc. is defined for each user and helps eliminate uninteresting links. This is typical of shared environments and environments where the system has to help a user (a lost user for instance) select a link [Tomek and Maurer 1992]. These problems make static links inadequate. It is therefore desirable for the system to automatically generate links on demand for the document author and the reader. Virtual linking is one solution. Virtual linking attempts to separate a link from its member nodes by delegating link generation and maintenance to the system at run time. Often these are links which cannot be pre-defined [Bieber 1992].

Several models have been proposed to support virtual linking mechanisms. They can generally be classified as employing history techniques, database techniques and information retrieval techniques. These three techniques are discussed below:

a. **History techniques** analyse the current static link structures or traversals to determine virtual links [Tomek and Maurer 92, Botafogo and Shneiderman 1991, Gloor 1991]. The philosophy behind this technique is that users are interested or not interested in nodes they have visited before. The inevitable change in user interests or browse patterns imply that *'there is no single optimal strategy'* [Tomek and Maurer 1992]. [Tomek and Maurer 1992] have also proposed ranking links according to some criteria based on user interest in certain nodes. They list several approaches to ranking links: random,

frequency of use, time stamps and media. Again for this strategy, no optimal strategy exists as none of these may be directly related to user-interest. The need for virtual linking is therefore paramount in both cases. The cognitive load required for navigation (e.g. HyperCard's history dialog box is therefore reduced).

b. **Database techniques** model associations by separating document and link (index) spaces [Frisse and Cousin 1989]. Proposals employing database models (e.g. the relational model [Hara et al 1991]) and a set theoretical approach [van Dyke 1991] have been made.

c. **Information retrieval (IR) techniques** can also generate virtual links [Rada 1991, Kotteman et a 1991]. The differences between database and information retrieval is:

'IR employs partial (vs. exact) matching; they are built on an underlying probabilistic (vs. deterministic) model; they classify information in a polythetic (vs. monothetic) basis; and queries are incomplete (vs. completely) specified' [Kotteman et al 1991].

These may use text scanning or attribute/value techniques. Text scanning facilities often utilise inverted files [Salton 1989, Bernstein et al 1991] to produce the links between textual nodes. Inverted files keep a record in an inverted list of documents that use a specific keyword and queries can be implemented by performing set operations in these lists. Attribute/value pair techniques attach descriptive attributes to nodes and links. The virtual link is then established by exploiting attribute/value pairs to match information e.g. Neptune [Delisle and Schwartz 1986], Grolier Guides [Laurel 1991] and StrathTutor [Mayes et al 1988].

3.3. Browsers.

Browsing is fundamental to the operations of a hypermedia environment. Browsers exploit the traceability property (see 3.2(c)) of links to traverse the nodes of a docuverse. Three types of browsers can be uniquely identified [Conklin and Begeman 1988, Delisle and Schwartz 1986]:

a. Node browser: A node browser allows node to node browsing, displaying information within the destination node. Some nodes are singular (e.g. contain only one information unit) while others are compound (e.g. contain lists). This term is more appropriate to the latter type, hence it is felt that

'context browser' is a more suitable term as the nodes referred to in this definition are actually contexts (cf. 3.1(b)).

b. Link browser: A link browser (document browser in Neptune) allows the user to browse following the links. A typical link browser is a path browser (a default route through the docuverse using an ordered list of nodes). It is intended to relieve the user of navigational duties.

c. Graphic browser: The graphic browser itself is a node containing a global diagram of the network of nodes. This can be used to orient oneself towards a specific node with respect to context or environment thus alleviating the much publicised problem of user-disorientation or 'getting lost in hyperspace'.

Inevitably, browsing may require synchronicity e.g. synchronising audio and video. In such cases tracing a link may prompt a number of actions on the appropriate nodes. Other useful features of a browser are filtering and animation. A node may be an aggregation of sub-nodes some of which may have nothing to do with the source of a link. Filtering ensures that only the relevant information is displayed when a node is visited. This technique may also be exploited to reduce information overload and to achieve fast scanning. Animation gives a visual impression of the trace e.g. tracing a generalisation hierarchy may be accompanied by an animation that gives the impression of an upward or downward transition.

4. Choosing a Hypermedia Environment.

The ill-structured nature of database design demanded that the project needed a general-purpose hypermedia system (c.f. 2(b) above) that would support:

a. management of data differing widely in structure, size and form- from text to graphics.

b. programmability: to offer the flexibility for interactivity and instantiation of virtual links among other features.

c. node and link definition, manipulation and presentation.

The hypermedia development system chosen for the project is HyperCard [Apple 1990, Goodman 1990]. HyperCard was chosen because it provides most of the above-mentioned requirements and is readily available:

HyperCard is a personal toolkit that gives users the power to ***use, customise*** and ***create*** *new information using...* ***text, graphics, video, animation, music, voice'*** [Apple 1990].

HyperCard is also widely available. In fact, Apples' bundling of HyperCard with every new Macintosh originally aroused much public interest in hypermedia.

HyperCard is a powerful system which fully exploits the user-interface facilities offered by the Macintosh. However, it is not a complete hypermedia system. Audio and video manipulation is attainable only through third-party add-ons such as SoundEdit [Farollon 1990]. Therefore some people have argued that HyperCard should not be taken as representative of hypermedia [McCall 1988]. However, the hypermedia facilities it readily provides, i.e. text and graphics, are deemed sufficient for the scope of this work. Customisation is very flexible because it is programmable via an easy-to-use English-language-based scripting language called HyperTalk [Winkler and Kamins 1990] which is part procedural and part object-oriented. It also support calls to external modules or applications. Another platform briefly considered was SuperCard [Silicon Beach Inc. 1989]. It is more like HyperCard but its performance under low-memory conditions was disappointing at the time the project started.

The following sub-sections briefly describe the facilities of HyperCard and give an overview of HyperTalk.

4.1. Node, Link, Text and Graphic Facilities of HyperCard.

This section describes the node, linking, browsing and media facilities of HyperCard as shown in Figure 3.2. No description is given of audio and video facilities as these depend on add-on products.

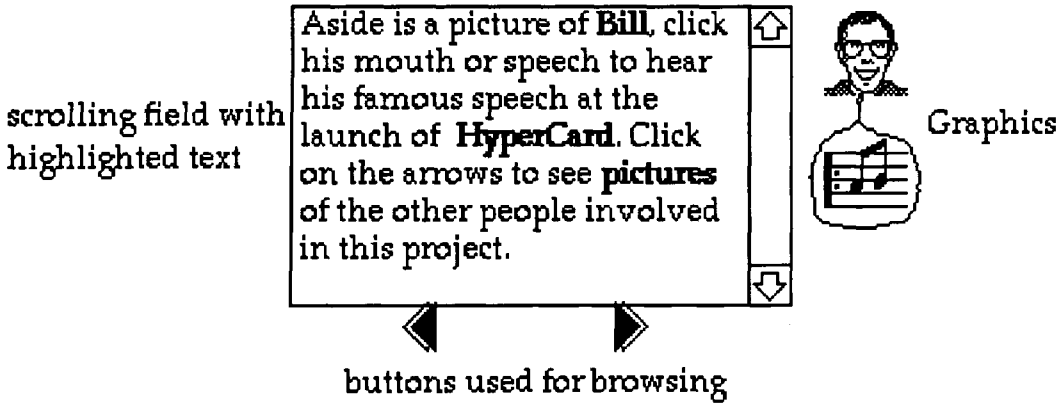

Figure 3.2: An Illustration of HyperCard Objects.

a. Node Facilities.

HyperCard stores information in nodes known as cards. A group of cards residing in the same file is collectively referred to as a **stack.** Additionally, a card can contain various combinations of text, graphics and links to audio or video data stored in one or more of the following HyperCard objects: **stack, background, card, field and button**. Each object is capable of receiving and generating events and this is mechanised through a script (text node comprising a series of program instructions).

b. Linking facilities.

The basic linking mechanism is provided by some activity incorporated into a **button** by means of a script. A card can contain buttons of the following types: transparent, rectangle, round-rectangle, radio, check-box. Buttons can also be passive, in such cases they can be used for presentation purposes e.g. labelling. A button can be named, and the name can be optionally displayed e.g. the name of an entity in an EER diagram. A button can further be associated with an icon (a graphic symbol) e.g. a cardinality/optionality icon for a relationship in an EER diagram. This encapsulation of a graphic (the button outline) and text (the button name) makes authoring of such nodes easy: the author just states the presentation without concern to how it is achieved.

c. Text facilities.

HyperCard stores text in information areas called **fields.** A field can contain text only. Fields have various useful properties, for instance, a field can be formatted to contain text of various fonts. The ability to format a field allows sections of text to be highlighted thereby representing an important concept such as an anchor. Also fields are typed, i.e., fields can be any one of the following types : **transparent, rectangle, opaque, shadow, scrolling.** The typing of fields has various benefits, for instance, it can store up to 32K of text, can be scrolling and can be programmable thus making the field a multi-link node. The scrolling field type seems most useful: text can be added to it without the need to create any additional nodes that are supposed to hold it.

d. Graphic facilities.

Each HyperCard object is a graphic entity. By changing the properties of a button or field a number of graphic shapes may be realised. In addition HyperCard provides conventional graphic features in the form of several painting tools (e.g. oval) and graphic operators (e.g. rotate) to paint various

shapes (lines, rectangle, circles, ovals etc.) on cards and backgrounds. However, under severe memory conditions these tools cannot be invoked.

4.2. Modes of HyperCard.

HyperCard has the following modes of operation: **browsing, typing, painting, authoring** and **scripting**. Modes are organised in a hierarchy of increasing functionality.

a. **Browsing**: Lets users look through stacks without the ability to modify stacks.
b. **Typing**: allows the addition of new cards and changes to text in fields.
c. **Painting**: The painting mode contains a number of tools and utilities used to paint and format pictures onto cards/backgrounds.
d. **Authoring**: Allows linking.
e. **Scripting**: Mode in which one can examine and modify the scripts that execute when messages are sent to certain objects. Stacks, backgrounds, cards, buttons and fields can all have scripts containing message handlers (see next section).

The capabilities provided in each mode are available in each succeeding node i.e. the scripting mode offers all capabilities.

HyperCard is also modal with respect to tool facilities. Browsing, button, field and paint operations are done through selecting a tool. For instance the creation of a button takes place with the button tool. While this may reduce the likelihood of accidentally editing some objects, the memory requirements for each often results in functionality to be compromised. In particular, it may be impossible to choose any one of the many painting tools if the computer does not have enough memory. This is a major setback for an author who endeavours to use transparencies. For instance one would want to trace transparent fields or buttons, and use a paintbrush tool to trace them and their grouping outline. This limitation often affects the functionality of the FD diagrammer which utilises such transparencies and also the size of the page (card) for FD diagrams is greatly reduced.

4.3. HyperTalk overview.

HyperTalk, the scripting language of HyperCard, is a command set comprising control structures, functions, properties, constants and operators to manipulate HyperCard objects. HyperTalk scripts are event-oriented in that they execute when certain events occur. Events can be internally or externally generated e.g. script-triggered events and mouse-clicks respectively. When an

event occurs, it ripples through HyperCard as a message which is executed when it finds an object containing its handler. Messages are passed according to a message-passing-chain: button/field > card > background > stack > HyperCard system (Figure 3.4). Since it operates on the basis of objects which interact by sending messages to each other, and also since a message can be generated by one object and executed by another higher level object, HyperCard exhibits some of the features of an object-oriented system. Message-handlers are equivalent to *methods* in object-oriented systems and the message-passing-chain is equivalent to an *inheritance* path. However, because of the inability to define new classes of objects HyperCard is not a fully-fledged object-oriented system.

The script is an important component of nodes. For example a script, belonging to a link button, that traces the link may look as follows:

```
on mouseup
   go to card destNode of stack stackName
end mouseup.
```

This script instructs the button that if it (the button) receives a mouse click (the mouseup event) then it goes to a specific card on the named stack.

5. Hypermedia: A Suitable Environment for Database Design?

Hypermedia has been used as the underlying technology for CAD (Computer-Aided Design) and CAISE in areas such as design, documentation, project management, analysis and software development [Delisle and Schwartz 1986, Bigelow 1988, Garg and Scacchi 1988, Cybulski and Reed 1992]. This use can be seen to have arisen from the inherent shortcomings of traditional databases in supporting CAISE, namely the use of fixed-format records and lack of support for version and configuration management.

The application of hypermedia as a storage model for CAD systems was propounded in Neptune [Delisle and Schwartz 1986], a prototype hypermedia system. Neptune also proposed a layered architecture, HAM (Hypertext Abstract Machine), for a generic application-independent model for the storage and access of nodes and links, contexts, distribution and synchronisation over a network. Nodes, links and contexts allow the abstraction mechanisms of hypermedia [Garg 1988] to be realised. Dynamic Design [Bigelow 1988] utilises the HAM architecture by storing requirements specifications, design notes, implementation notes, code (source and object), test data etc. in nodes. Associations between nodes are manifested in the form of attributed links showing the type of the link (e.g. relatesTo, leadsTo). Contexts are used to

realise local and global workspaces as well as configuration management. DIF [Garg and Scacchi 1988] is purely a documentation and browsing system with links to other tools in the System Factory Project [Scacchi 1989].

Recently, [Cybulski and Reed 1992] described the integration of CASE tools in an extended hypermedia environment called HyperCASE. HyperCASE is a knowledge-based hypermedia document repository. Figure 3.5 is an extract from their work showing navigation among software documents and hypermedia applications to CAISE. Figure 3.5 is indicative material suggesting that hypermedia is a suitable technology for the authoring and management of software development documents.

Schema modelling (c.f. Chapter 2, section 2) utilising the entity-relationship (E-R) model is an area well-suited to the application of hypermedia. Each entity may be represented by a node in a hyperdocument. The node may be populated with information such as the entity's properties, descriptions and annotations. Links may be used to represent entity-relationships. The link can be embedded within the entity nodes or represented in a link node that may be further populated with additional information about the relationship.

Organising information as nodes goes a long way to address the problems associated with incompleteness and iteration in database design. Incremental design is conceivable as the data analyst judiciously adds or edits nodes, as well as capturing information as it becomes available. Further, by judiciously articulating links (relationships) the data analyst can choose to represent data in ways that are more communicable to interested parties.

The same environment could be used to produce the E-R diagrams shown in Figure 3.5 (c.f. Chapter 2, Section 4.2a(ii)). E-R diagrams employ a graphic notation which utilises symbols (rectangles, arcs, icons) to represent entities, relationships and their constraints, and may also include the attributes of entities and relationships. The symbols are optionally named, manifesting a mixture of text and graphics. A composed node (see 3.2 (a)) comprising text (for the name) and graphic (for the symbol) can be formed. Organising E-R design information as nodes in this way offers opportunities for interactive diagram editing, view filtering and automatic linking of diagram nodes. Database designers may then browse through the E-R model following the links. In general, hypermedia is useful both in presenting the E-R models as well as in dispensing the models.

The investigation of the applicability of hypermedia to database design can also be extended to the process of schema integration (c.f. Chapter 2, Section 5). Individually produced subschemas can be integrated in a controlled manner to

form a conceptual schema in a hypermedia environment that links the input concepts, the resultant integrated concepts and any related information. Indeed, a collaborative dimension can be added to the environment to form a framework for deliberation with the aim of capturing the rationale for integration decisions. The proposed environment therefore utilises hypermedia for problem-resolution, literary-exchange and documentation (c.f. Section 3.2) in the data modelling domain. Such an environment would establish an integration knowledge-base that is invaluable during database maintenance as well as communicating expertise to data analysts not yet experienced with the organisation's data. It may further help in the dimension of business analysis by giving indications of areas of duplication which management might be happy to eliminate.

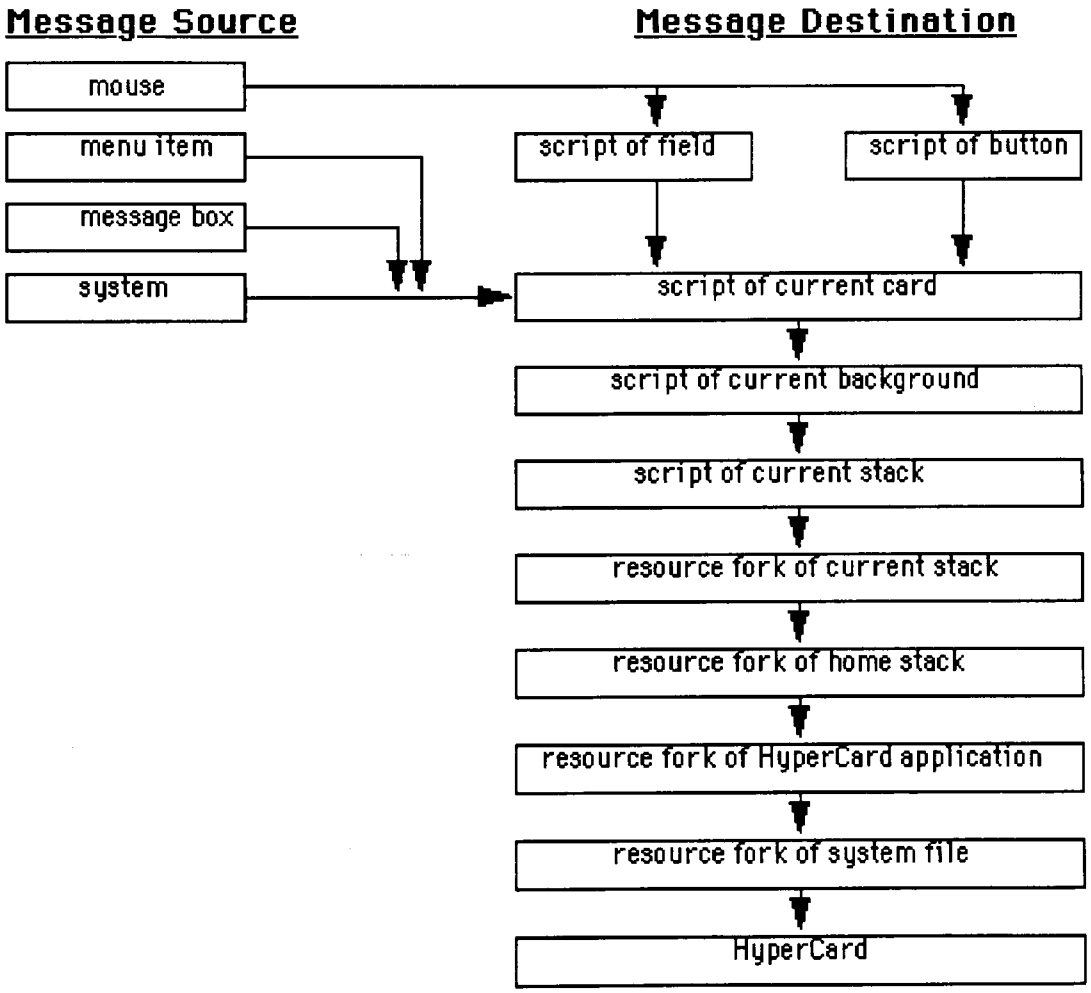

Figure 3.4: HyperCard Message-Passing Chain [Williams 1987].

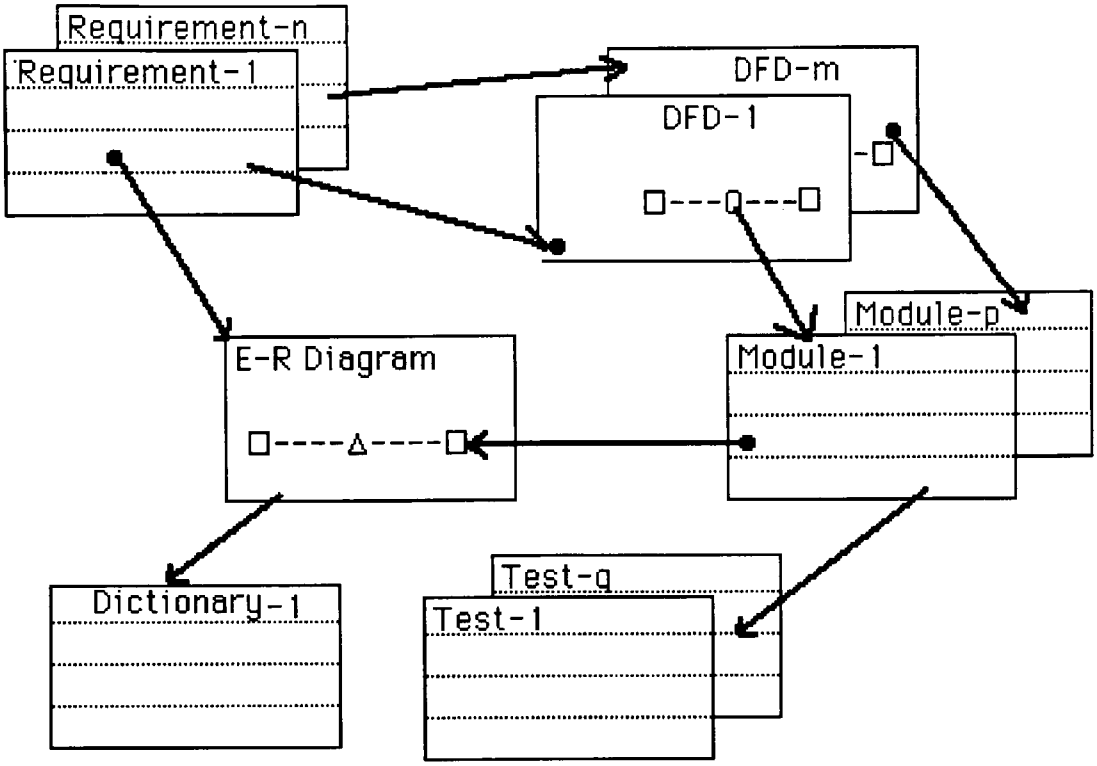

Figure 3.5: Navigation Among Software Development Documents.

6. Conclusion.

This chapter provided an overview of hypermedia. We outlined its historical development and its architecture. A detailed discussion was given of the major architectural components of hypermedia systems: nodes and links. We also discussed how these components have been exploited by some sample representative hypermedia systems classified in terms of the following areas: on-line browsing systems, literary-exchange systems, problem-resolution systems and general-purpose systems. The major theme of the chapter is the use of hypermedia as an environment for CAISE. The chapter concluded by discussing how hypermedia renders itself suitable to schema modelling and collaborative schema integration.

The next chapter discusses a prototype tool developed to encapsulate the key features of schema modelling. Schema integration issues will be discussed in chapter 7 where the collaborative schema modelling tool is described.

A Hypermedia Environment for Schema Modelling and Documentation.

Summary.

This chapter discusses HSMS (Hypermedia Schema Modelling System) tool. Section 1 is an introductory section briefly discussing the environment, application domain and operational nature of the tool. Section 2 discusses the architecture of HSMS. Section 3 presents the integrated entity/relationship dictionary facility around which the functionality of the whole tool is centred. Section 4 discusses the F-D and E-R diagramming facilities. The diagramming mechanisms are presented, as are the 'entity-clustering' features of the E-R diagramming facility. The links between the dictionary facility and the diagramming facility are also given in the discussion. Facilities offering localised linking, querying, integrity management and browsing are discussed in section 5. Section 6 gives an overall view of the schema modelling tool from the perspective of hypermedia technology. Section 7 concludes the chapter.

1. Introduction.

A case for utilising hypermedia for schema modelling was presented in chapter 3. In summary, the following areas of application were identified: design process support, documentation and logical document usage (reading). In particular, hypermedia was tentatively proposed for representation, presentation and management of design documents of the schema modelling process. The development and evaluation of HSMS to demonstrate the salient features of the proposed database design environment are the major themes of this chapter.

HSMS is a schema modelling tool developed as a major part of this research project. It runs on Apple's HyperCard [Goodman 1990] and was developed in HyperCard's scripting language HyperTalk [Winkler and Kamins 1990]. It also consists of a suite of subservient modules developed in Symantec's Think Pascal [Symantec Corp. 1990]. Its domain is conceptual modelling for standard information systems, using the extended E-R model as the data model. The design artifacts of the extended E-R model [Teorey et al 1986] are attributes, entities and association, generalisation and aggregation relationships. Dependency, optionality and cardinality constraints defined upon the model are also included. Each of these constructs is represented and organised as a node by the tool (backgrounds, cards, fields, buttons) (cf.

Chapter 3, Section 4.1). HSMS is therefore both a modelling and documentation environment. The next section discusses the architecture of HSMS.

2. Architecture of HSMS.

Figure 4.1 shows the 4 major facilities of HSMS, namely: the schema manager, the dictionary, the E-R diagrammer and a suite of utility modules. The schema manager (component 1 in the figure) is that part of the tool responsible for the creation, opening and deletion of schema stacks. It offers the database designer a list of menus to create, open or delete a named schema. It also ensures that all schemas are named uniquely and appropriately and that no schemas are accidentally deleted.

The dictionary facility is a repository of the conceptual objects of the design space. In particular, it records information pertaining to entities and relationships (component 2 in the figure). The E-R and F-D diagramming facilities (component 3 in the figure) are used to produce diagrams based on the specified relationships between the entities in the data space, or dependencies between attributes of an entity or relationship respectively. The totality of the dictionaries and the E-R diagrams constitute a hyperdocument representing the whole schema. The utility modules (component 4 in the figure) are a suite of complementary facilities composed of browsing, querying, linking and critiquing facilities. The arrows in the figure indicate the directions of data interchange between the various facilities.

All components of HSMS utilise a graphical interface. The graphical interface heavily utilises menus, dialog boxes and icons. Dialog boxes are special kinds of menus that appear in a window despite the state of the mouse, unlike conventional menus which appear for as long as the mouse is pressed. Icons are graphical symbols used to support a graphic notation or to symbolise functionality.

3. The Dictionary Subsystem.

The dictionary subsystem comprises an integrated entity/relationship dictionary. This integration is provided because of the tight coupling between entities and relationships in the design space. For every schema there is only one dictionary and for every dictionary there is only one schema. Each dictionary entry is either an entity or a relationship and each entry is encoded in one node (card). Each node holds information about the intrinsic properties of a concept. This information is represented in sub-nodes of the

node. Entities or relationships are in turn grouped together under the same organising context (node). This means that before a concept enters the dictionary, it has to be recognised and classified as an entity, a relationship or an attribute. These three concepts are discussed below.

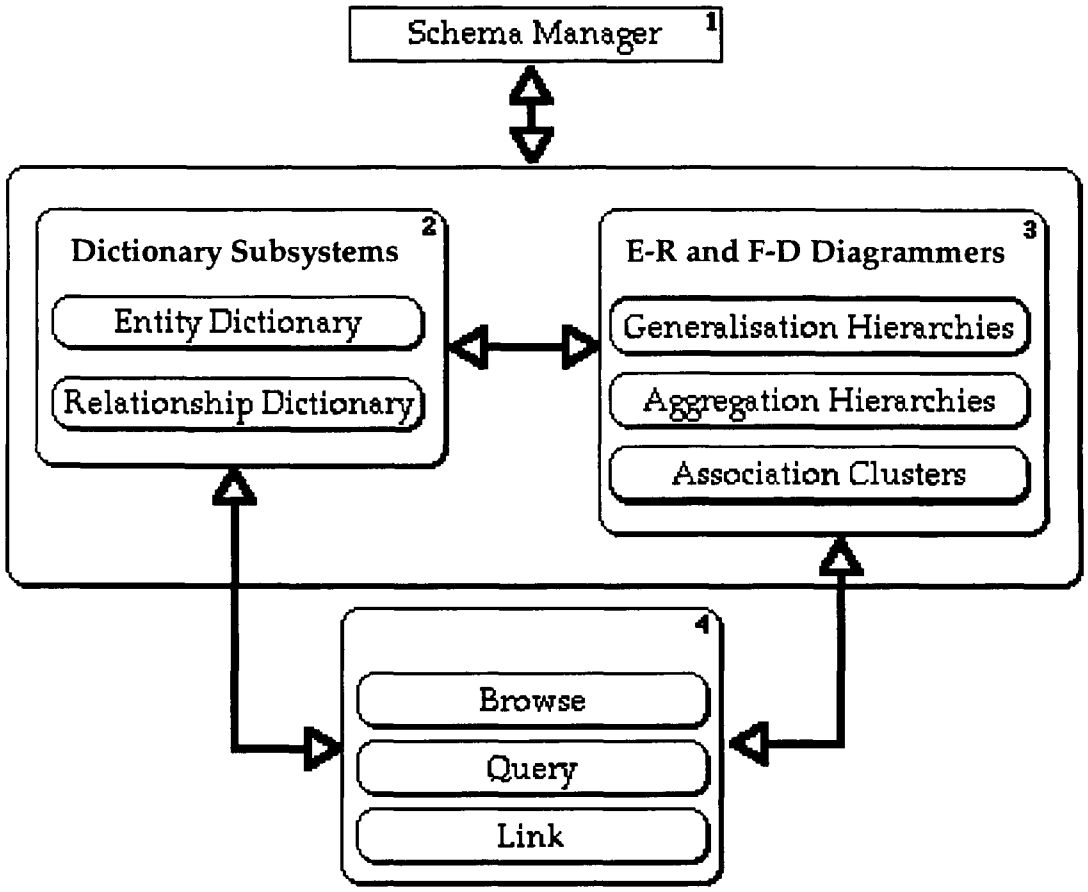

Figure 4.1: Architecture of HSMS.

3.1. Concept recognition.

A concept is defined as anything recognised as of interest to the data processing needs of an organisation. Concepts are both linguistic and conceptual abstractions. Linguistic abstractions are names given to the concepts. Names define a terminology for the design space and as such can be considered as the higher abstractions that help to describe or predict reality [Sowa 1984]. Given the name of a concept, a knowledgeable user/designer can better approximate its structure and behaviour. Conceptual abstractions establish the intrinsic properties of a concept. These include its structure (attributes), behaviour (constraints defined upon the concept) and its description.

3.2. Concept classification.

A concept in the extended E-R model is classified as an entity, a relationship or an attribute. Classification in terms of these types is subjective, depending on the perspective of the user/designer [Beynon-Davies 1992(b), Baskerville 1993, Howe 1993].

The major processes supported by the dictionary subsystem are editing (section 3.3), linking (section 5.1), querying (section 5.2), integrity management (section 5.3) and browsing (section 5.4).

3.3. Editing.

A basic facility that a database designer needs is the ability to create and define entities and relationships. An editing process involves representing a given concept in some computer medium. Concepts are edited in textual form via a structured graphical interface i.e. an interface with pre-defined templates. The advantages of a structured interface are twofold:

a. The editors are context dependent i.e. their behaviour depends on the type and status of the node being edited. Once an entry has been saved, its name cannot be edited directly. Instead, the designer has to resort to the delete or rename facilities provided. However, descriptions can be edited at any time, as can an attribute's type, and any internal constraints. The designer is notified of any implications of changes to the type or constraints of prime attributes to the defined key of the entity and/or relationship (cf. consistency and completeness).
b. The interfaces are hierarchically-organised to depict the abstraction levels. Figures 4.2(ii) and 4.3 show how the abstractions are hierarchically organised in a top down fashion. Further, data filtering can be achieved in a natural hierarchical fashion. Figures 4.2(i) and 4.2(ii) show how the level of detailed displayed on an entity node can be changed from detailed to more abstract, and vice versa.

The editors offer a designer the ability to edit entities/relationships. As shown in figures 4.2(ii) and 4.3 the nodes behind the interfaces are populated with the following information:

a. a one-word **entity/relationship name** entered by typing or selection in the case of abstraction relationships. A validation procedure excludes illegal

characters (e.g. space) during typing. For relationships, the member entities are selected from a list of defined entities listed in a pop-up menu.

b. a free-text **entity/relationship description**.

c. a one-word **attribute name** (validation same as for entity name).

d. the **data type** of the attribute (selected by means of a pop-up menu of data types- customisable to include user-defined types). The initial list of data types consists of: text, boolean, character, integer and real, and the default is character. The designer can optionally provide a description for a user-defined type as well as requesting descriptions for all stated types.

e. the **variability** of the attribute's value for any entity or relationship instance. The assumption is that attribute values that never change during the life of an entity or relationship instance may be prime i.e. may be part of the key.

f. the **optionality** of the attribute for any entity or relationship instance. Non-optional attributes may be prime since they are never null.

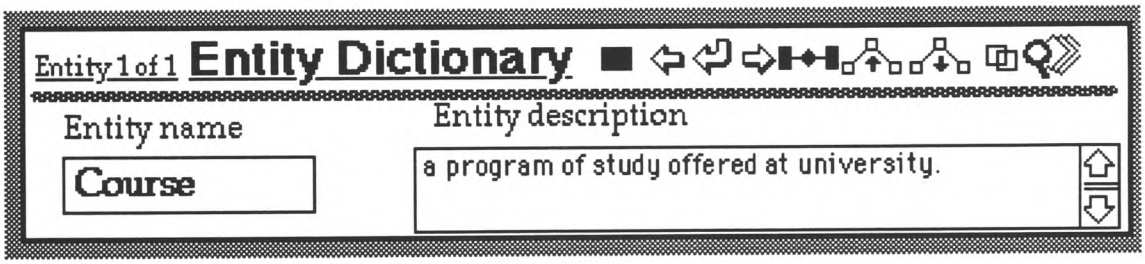

Figure 4.2(i): An Illustration of a Filtered Entity.

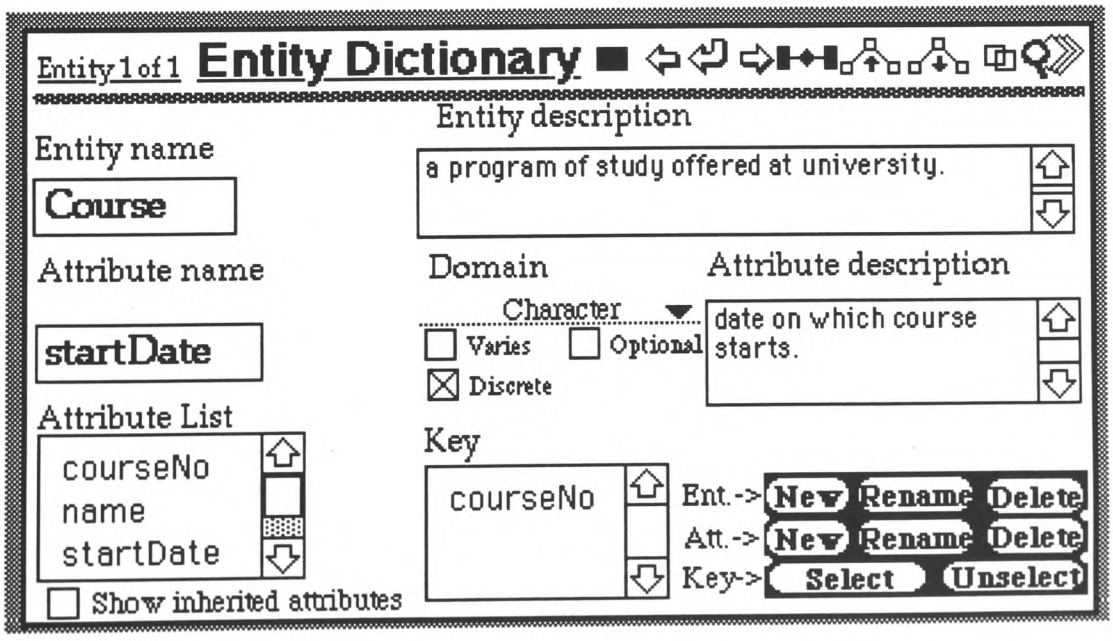

Figure 4.2(ii): An Illustration of a Detailed Entity.

g. the **discreteness** of the attribute's values: an indication of whether the domain of the attribute is countable i.e. the domain can be put into a one-to-one correspondence with the set of integers. The underlying assumption is that an attribute that is not discrete (such as a real number) is a bad candidate for a key.

h. a free-text **attribute description**.

i. the **key**: selected from a list of defined attributes. The key definition facility validates attribute selections based on the status specified in (e), (f) and (g) above. The defaults are 'non-variable, 'discrete' and 'non-optional'. For association relationships additional entries are required to define relationship constraints. The defaults are: 'one-many' cardinality and 'mandatory-mandatory' optionality.

j. an **attribute list**: the list of currently specified attributes. To get details about any one of the listed attributes, the designer clicks on the attribute. All the information pertaining to the attribute is then displayed in the attribute template.

k. **browsing tools:** a number of browsing tools are provided (c.f. Section 5.4).

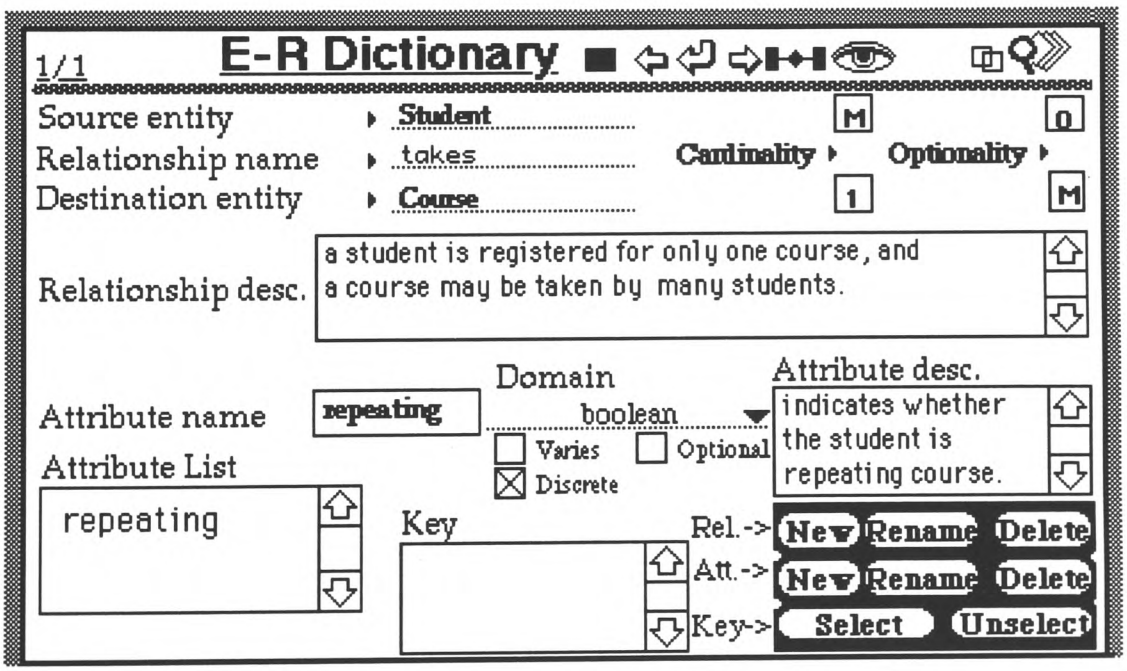

Figure 4.3: An Illustration of a Detailed E-R Entry.

The node representing an entity or a relationship is named after the entity or relationship. These nodes (and relationships) are created, deleted and renamed via icons that appear at the bottom-left of the screen. Most of the

sub-nodes are pre-defined on the templates. The designer edits them by clicking the mouse on the sub-nodes and depending on the edit state of the target sub-node, editing may proceed by typing or by selecting from a pop-up menu.

4. The Diagramming Facilities.

The use of structured diagramming techniques has long been recognised and established for database design work [Martin and McClure 1985]. The problems associated with the manual production, documentation and version control of such diagrams has been equally recognised. This has led to the CASE community developing various drawing tools. The E-R approach has been particularly subject to attention. In particular, a number of tools have been developed for E-R diagramming: Deft [Sysbase], Select SSADM [Select Software Tools 1993], Excelerator [Index Technology Corp. 1987, Williams 1988].

Producing good E-R diagrams demands a suitable environment for their production and management. Foremost, an environment capable of presenting the graphical symbols of a notation in use [(Chen 1976 and Bachman 1969]), Martin 1989] is required. In addition, the diagramming tool has to offer editing operators, consistency and completeness checking and tool tailoring [Martin 1988].

This section discusses two diagramming tools of HSMS: HFDD and HEERD. HFDD (Hypermedia Functional Dependency Diagrammer) is that tool used for the production of hyperdocuments of functional dependency diagrams. In contrast, HEERD (Hypermedia Extended Entity-Relationship Diagrammer) is used for producing hyperdocuments of extended E-R diagrams.

Prior to diagram editing, the relevant information has to be transferred from the nodes of the data dictionary to the target node (stack) of the diagrams. Each time a transition is made from the dictionary to the F-D or E-R stack, information is automatically collected from specific nodes of the dictionaries (use of typed nodes) and the relevant repositories in the diagram stacks are updated. A transition to the F-D stacks collects only attribute information, while that to the E-R stacks collects entity and relationship information.

Figures 4.4 and 4.5 illustrate diagrams produced by HFDD and HERRD respectively. The major difference between the two diagrammers is that the latter is targeted at higher level relationships (i.e. external ties between

entities) while the former is targeted at the internal ties between the attributes of an entity or relationship. F-D diagrams complement the data dictionary by representing the logic for key selection and/or providing intra-entity/relationship semantics. It is therefore anticipated that HFDD be employed earlier in the design process. Hence it is introduced before a discussion of the E-R diagrammer which preys on data dictionary information supplemented by HFDD.

4.1. The F-D diagrammer (HFDD).

In addition to being non-variable, discrete and non-optional, a key must be unique to differentiate an instance of one entity or relationship from another. Implicit in the definition of a key is the fact that all attributes of an entity/relationship are dependent (i.e. uniquely identified by) on the key. Key identification is based on an analysis of the functional dependencies amongst the attributes of the entity or relationship. A key is a set of attributes. It can be simple (composed of one attribute) or compound (composed of more than one attribute). Such dependencies are usually either not clear or are taken for granted. It is hoped that by explicitly defining keys, the logic for key selection will be more communicable. This is especially true of the cases where the key is compound, where there is more than one candidate key or the relationship has its own attributes, some of which are potential keys.

In Figure 4.4, the F-D diagram explicitly shows how the compound attribute 'student.studNo, course.courseNo' uniquely determines the attribute 'takes.grade'. It also shows how the attribute 'course.courseNo' uniquely determines the number of students enrolled for the course.

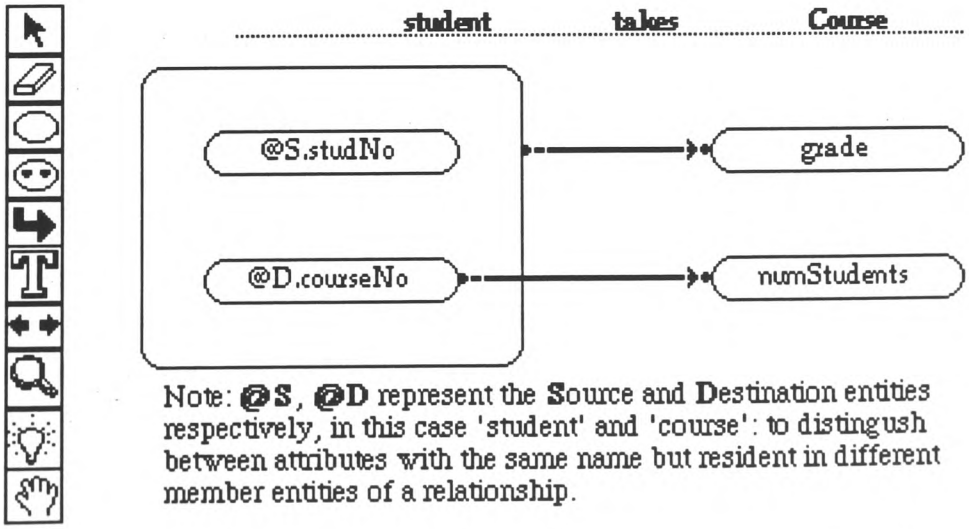

Figure 4.4: An Intra-Relationship F-D Diagram for 'Student takes Course'.

The major operation available in the F-D diagrammer is editing. Each F-D diagram resides on one node (card) and each card holds an F-D diagram for only one entity or relationship. At the top of each diagram, is the name of the entity or relationship with that dependency diagram. To edit an F-D diagram, the data modeller:

a. Chooses an appropriate edit tool from the edit palette shown on the left in Figure 4.4. The edit tools (in top to bottom order in the figure) are as follows:

   i. a general purpose 'selection tool' used to select a diagram symbol.
   ii. an 'erase tool' used to erase a selected diagram symbol.
   iii. an 'attribute tool' used to select and create an attribute symbol.
   iv. a 'compound tool' used to group attributes (selected in Figure 4.4).
   v. a 'dependency tool' used to draw F-D arcs between two attribute symbols.
   vi. a 'text tool' used to edit diagram descriptions.
   vii. a 'browse tool' used to go to the previous or next diagram (page).
   viii. a 'find tool' used to locate and visit the diagram containing a selected attribute.
   ix. a 'zoom tool' used to go to the dictionary entry of a selected attribute.

b. Clicks on a diagram symbol to effect the edit operator. The operation actioned will depend on the edit tool chosen and the type of the target artifact (use of typed nodes). The resultant operations include:

   i. The creation and positioning of a new attribute. A grid is pre-defined which defines regions on the page in which simple attributes can reside. To position an attribute, the designer moves to and clicks the mouse at the desired page location. The tool then checks whether the located attribute overlaps other attributes or dependency arcs. If so, the user is prompted and the operation is undone.
   ii. The tracing of a dependency arc. HFDD monitors the trace and draws an outline of the arc region obeying the directional constraint that restrict arc segments in the vertical or horizontal dimensions. The segment is always defined by the current mouse location and the point of inception of the segment. After sketching the arc path, the terminal points of the arc are aligned to the allowed attachment points on the attribute. The dependency arc is then drawn with the appropriate directional notation.

iii. the deletion of the selected symbol. Deletion of an attribute leads to the deletion of all dependencies involving the attribute.

iv. the relocation of a diagram object. Attribute relocation leads to a partial relocation of the dependency-arc segments emanating from it. Dependency arc relocation is achieved by selecting and relocating the individual arc segments which may lead to readjustments of any segments adjacent to it. The relocation preserves the directional constraints on an arc segment unless the relocation is such that two arc segments have been merged. Checks are also done on overlaps.

v. pressing the mouse over a dependency constraint icon pops up a menu from which a new constraint icon is selected, thus modifying the semantics of the dependency.

4.2. The Extended E-R diagrammer (HEERD).

HEERD is the more elaborate of the two diagrammers since E-R diagrams are the major deliverables of the conceptual modelling process [Martin and McLure 1985]. HEERD offers a graphical equivalent to the E-R dictionary. It is a tool with which the designer can graphically create, present and manipulate the proposed database schema. Figure 4.5 shows a sample interface to the E-R diagrammer.

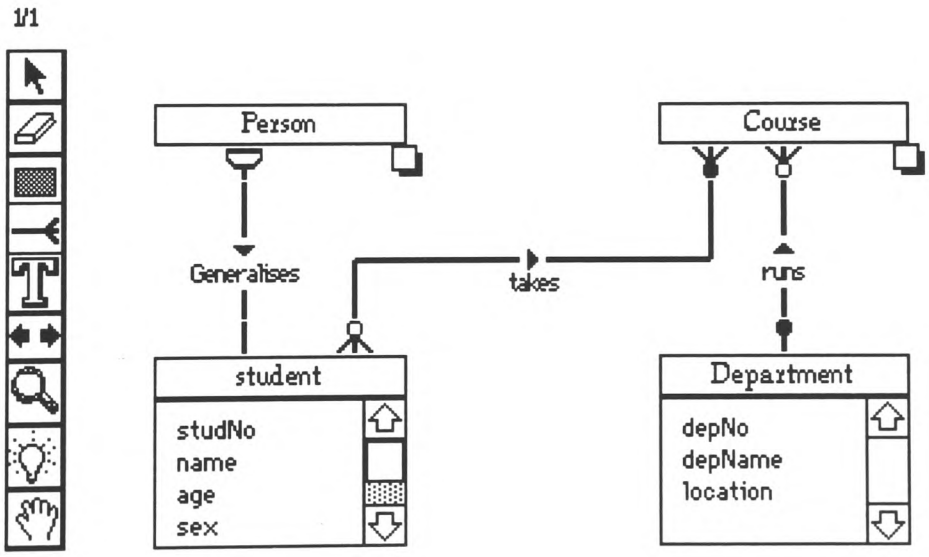

Figure 4.5: A Sample ER Diagram Drawn Using HEERD.

The editing operations of HEERD are basically the same at that of HFDD with a few differences. The 'grouper' tool is not present because the purpose it serves in HFDD is catered for in HEERD by generalisation and aggregation

relationships. In place of the 'attribute tool' is the 'entity' tool which actions the same operation. The 'find tool' and the 'zoom tool' operate on both entities and relationships. When a diagram entity node is created, however, an additional node that contains a list of the entity's attributes is attached to it. This attribute node can be closed or opened by clicking on it. Closing the node sometimes makes the diagram more communicable because of reduced detail. For example, in Figure 4.5, the attribute nodes for the entities 'Student' and 'Department' are displayed while those for the entities 'Person' and 'Course' are closed.

In place of the 'dependency tool' is the 'relationship' tool for drawing relationship arcs. When this tool is selected, a dialog box appears displaying the available relationships. After the selection, the designer proceeds as for dependencies. However, when it comes to drawing the relationship arc, the facility uses its knowledge about the defined constraints on the relationship in question and automatically sets the icons for the constraints appropriately. Unless the designer wants to modify the constraints of the relationship, there is therefore no need for the designer to interactively specify the relationship constraints on the diagram.

4.2.1. Clustering.

Entity clustering is the process of identifying a group of entities that satisfy some defined criteria. For example, entities in a schema may be clustered:

a. Structurally: entities with close structural relationships such as generalisation and aggregation are grouped together.
b. Logically: entities with close logical relationships (association relationships) are grouped together.
c. Information areas: entities are clustered together if they belong to the same information area of an organisation and/or according to the intention of the designer who produced them.
d. A combination of the above, possibly with weights assigned to the relationships existing.

Entity clustering leads to the production of otherwise small schemas. However, small schemas are both advantageous and disadvantageous during the database design process. Studies by [Miller 1956] showed that the human mind is capable of processing an average of seven objects at any time. [Feldman and Miller 1986] add that the communicative power of a diagram is

inversely proportional to the number of objects therein, and suggest that a diagram with more than 30 objects is not easy to understand. This implies that smaller schemas are easier to understand and process in the human mind and they are more likely to fit on the same diagram. In the schema integration sense this means that the schemas will be easier to merge and resolve because there are fewer conflicts. This, however, is true for schemas that are complete i.e. the whole schema is fitted on one face of the display media (e.g. one page) because processing of inter-document information will require some relational processing between the involved documents.

Faced with the problem that HyperCard supports variation of card (page) sizes to a limit, entity clustering is a reasonable way to try and split a large schema into smaller ones that can fit on a page (residing on a card). This inevitably leads to relationships between the pages, or rather the subschemas resident on those pages. This technique is in line with that adopted for DFDs in which a child diagram is created for a process. In this case a child schema is created for a schema and a corresponding child diagram is produced. Producing child diagrams requires the production of entity clusters first and the concept of closures is used for such purposes (see discussion under (a) below).

Thus the organisation themes of HERRD differ considerably from that of HFDD. Instead of a one-to-one correspondence between a diagram and an entity/relationship, HERRD uses clustering techniques to represent on the same diagram (node) a number of 'related' relationships. The clustering facility was motivated by two factors: one technical and the other social. The following subsections discuss them in turn.

a. Technical factor.

The size of a card (the node on which diagram objects are drawn) is restricted by the environment, HyperCard. Therefore the number of entities and relationships displayable on the card without overlaps is also restricted. A method had to be found to split a large diagram so as to fit parts of it onto the card area. The method had to satisfy two conditions:

i. **conform to a major aspect of the real world usage of the entities** forming the cluster. Relationships sharing entities form a good basis for clustering such entities.
ii. be **iterative** so that the method can be repeatedly applied to a cluster until a suitable number of entities remain.

The concept of closures (logical horizons) [Ellis 1982, Desai 1990] seemed attractive. The closure of an entity is defined as the set of entities uniquely identified by the given entity, directly or transitively. It can be found by iteratively following all 'to-one' relationships emanating from the (basis) entity. If there is such a closure, then the entity in question forms the basis of a cluster. Because of the interdependence amongst the entities, a closure may partially satisfy (i). (ii) can be satisfied by preferentially handling direct closures as opposed to transitive closures. Having found a suitable cluster, it can be allocated a page on which to draw its entities and relationships. It is worth noting at this stage that a relationship cannot exist in more than one closure, but entities can.

However, total reliance on this concept is not desirable as a closure may not exist, or if it does the resultant diagram may be sparse relative to the page area. Provisions have to be in place to add more diagram objects to a page (provided the space is there) and to create and dispose of pages containing diagrams.

b. Social Factor.

The communicability of a diagram depends, among other things, on the size of the diagram and how the diagram objects are arranged on the page [Martin and McClure 1985, Feldman and Miller 1986].

Before a relationship can be created, the tool therefore needs to know the page on which to draw it. This is done preemptively by a cluster selection utility which, when invoked, gives the user a list of entities around which clusters can be formed. Clusters are found on the basis of the relationship types: association, generalisation and aggregation. Having selected one of the entities, the user is presented with information about the cluster. This includes the entities and relationships in the cluster, and more importantly, any existing diagrams sharing some entities (and their number) with each one of these clusters. Given these measures of cluster cohesion and coupling, the designer has an option to go to a specific page or create a new page on which to draw the cluster's E-R diagram.

Further, a utility is provided to highlight clusters on a diagram. When invoked, this utility presents for entity selection a list of entities as above. The closure of the selected entity is computed, the involved relationships determined and the cluster is highlighted. This helps in moving clusters between pages as well as determining the 'cohesion' of a cluster.

5. Utilities.

A number of utility tools are offered by HSMS. They offer facilities for linking, querying, integrity management and browsing the design hyperdocuments. The linking mechanisms incorporate structural links (cf. section 5.1 (a)), virtual links (cf. section 5.1 (b)) and user-defined links (cf. section 5.1 (c)). The integrity management mechanisms incorporate data validation (cf. section 5.3 (a)) and design critiquing (cf. section 5.3 (b)).

5.1. Linking.

Links are probably the most important constructs of a hypermedia system [Tomek and Maurer 1992]. Links connect nodes and can span the whole HSMS hyperdocument. In addition, links offer an opportunity for pre-defined trajectories (structural links) through the hyperdocument as well as grouping related information into the same node (alternatively known as collections or contexts). Users may also want to define their own links (user-defined links) or even utilise links implied (virtual links) in the hyperdocument. These forms of links are prominent in HSMS due to the general nature of design work and they have a positive bearing on browsing and filtering.

HSMS uses three types of links: structural, user-defined and virtual. Figure 4.6 is a sample illustration of these links and the design artifacts they connect.

a. Structural Links.

Structural links are links established by the hypermedia engine as the various nodes are created. HSMS exploits and utilises these links as organisational and presentational themes in various ways. The ability to nest nodes (i.e. nodes residing on top of other nodes e.g. a field residing on a card) offers the possibility of organising data into contexts [Delisle and Schwartz 1987]. Contexts roughly correspond to classification constructs. All entities or relationships share a common context by virtue of residing in the same background. The same can be said about fields displaying a list of attributes.

Organising nodes as contexts goes a long way to reducing the effect of a major issue that hypermedia has been rightfully criticised for: disorientation [Rada 1991, Conklin 1987, Halasz 1988]. In HSMS, one is 'always' aware of one's orientation because the context within which to interpret any node is readily apparent. Further, in HSMS traversals are predictable and controlled (see Section 5.4) because the ability to switch contexts rests solely with the user. Hence the problem of 'getting lost in hyperspace' is greatly reduced. Also, selective exploration is possible i.e. choosing what is less understood or

important, or going up or down the abstraction gradient.

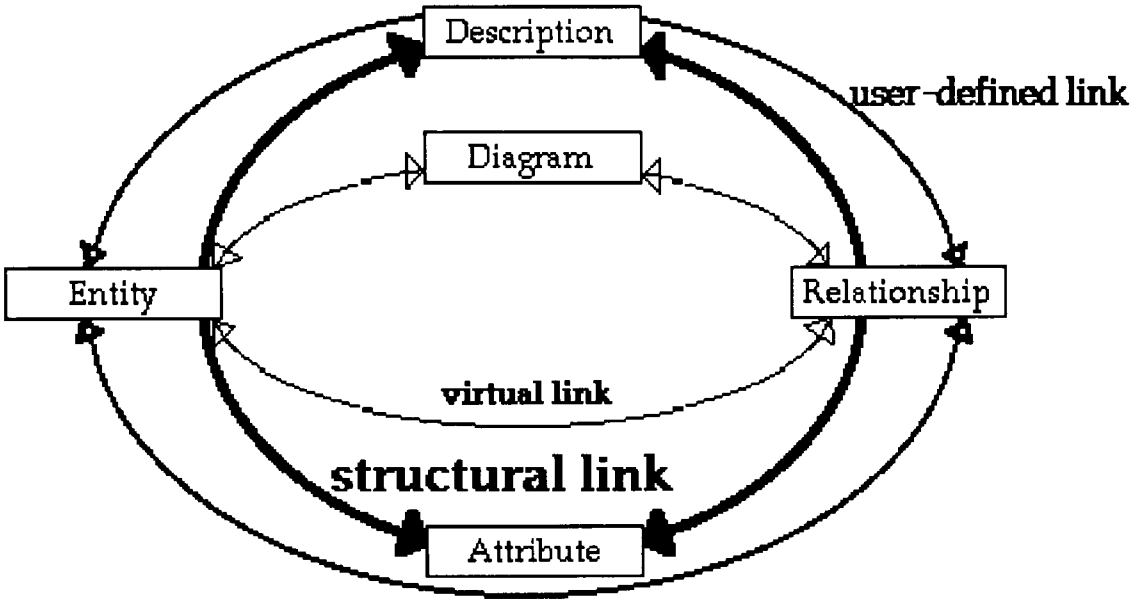

Figure 4.6: Links Between E-R Modelling Components.

b. Virtual Links.

Virtual links are vital in design work because some links may be implied. The dynamic nature of the design process is not well supported by 'static' (permanent) links. Some links need to be dynamically 'constructed' as modelling constructs are created, deleted, renamed or relocated. For instance, there is an implied link between a sub-entity and its ancestors, and in particular all the attributes and relationships of its ancestors. A static link between a dictionary entry and a diagram object cannot be guaranteed forever as the diagram is rearranged to maximise its communicability. HSMS attempts to resolve a virtual link by generating a specific instance of the link when required. In general, HSMS resolves virtual links by searching methods. For example, in generalisation, an inheritance link is resolved by upward traversal through the generalisation hierarchy.

c. User-defined links.

User-defined links are another useful group of links for data modelling. In HSMS, user-defined links are used to lead the user to some chunk of information related to the one in question, but otherwise not implied, virtually or structurally. User-defined links are normally used for linking free-form textual descriptions to a syntactic token. For example, the description for a student stated as 'person undertaking *programme of study*'

has the italicised text representing a user-defined link to the entity 'course'. As can be seen from this example, links involving free-form textual data require natural language processing capabilities. This may be achieved through analysing the text but this approach has socially-based problems of its own (c.f. automation in schema integration- synonyms and homonyms [Batini et al 1986]).

User-defined links in HSMS are created or deleted as follows:

i. The designer marks an anchor point (source or destination of link) by highlighting text and designating it as the source or destination of the link. A floating palette displaying the selection and its immediate neighbourhood appears on the screen.
ii. The designer browses through the hyperdocument in order to locate the other anchor point of the link. Another floating palette displaying the new anchor also appears on the screen.
iii. Finally, the text selections are linked/de-linked by issuing the menu command to link-up or unlink the anchors. The tool appropriately updates the repository of link information. The link is established and traversal can be performed from it.

However, the management of user-defined links is a complex process. The iterative nature of design work implies that the existence of an anchor (the source or destination selections) is not always guaranteed as design information is modified, deleted or moved. To avoid dangling links, HSMS 'intelligently' observes the edit operations performed on anchor data. Character insertions and deletions within an anchor lead to expansion and reduction of the anchors respectively. The use of floating palettes proved useful in creating links across stacks [Bieber 1992] and memory support by displaying in the palette information about the immediate neighbourhood of the link. Further, HSMS provides facilities to hide or show anchors.

5.2. Querying.

The query facility allows the designer to query dictionary information. Figure 4.7 show a diagram of the query interface. It has two sections, one for entity queries and the other for relationship queries. In both cases, the user is presented with a list of entities and relationships to choose from. A list of attributes of the selected concept is then displayed.

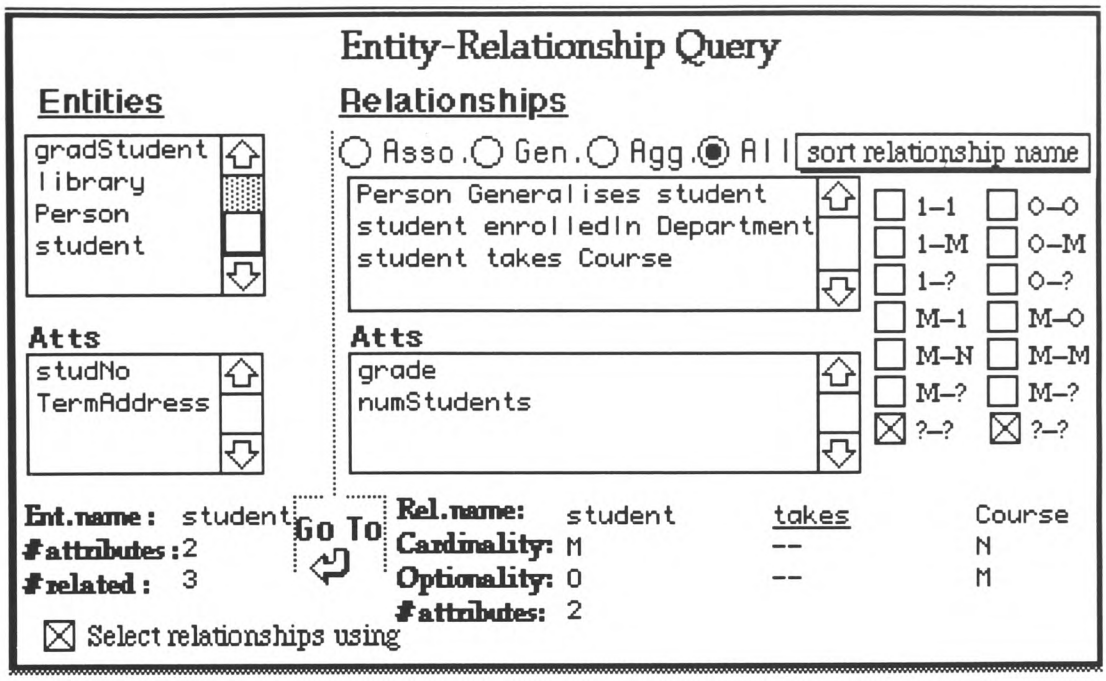

Figure 4.7: A Sample Entity/Relationship Query Formulation.

For relationships, the query criteria is defined prior to execution of the query. At the highest level, the designer formulates a query on all or a specific relationship type (association, generalisation or aggregation). Further, the search criteria may be limited to relationships conforming to specific constraints only. The designer may further restrict the search criteria by querying those relationships in which a selected entity is a member.

Figure 4.7 shows a query performed on the entity 'student'. It lists its attributes, their number and the number of entities it is related to. The search criteria has been restricted to 'all' those relationships involving the entity 'student' with any constraints as shown by the check-boxes in the diagram. Of these relationships, one is selected and the details displayed. One could, for instance, restrict the search criteria to all 'one-one' relationships.

5.3. Integrity Management.

The major mechanisms for integrity management are data validation and design critiquing.

a. Data validation.

Inconsistencies may manifest in a data dictionary when the data is entered. HSMS attempts to solve this problem by not allowing inconsistent data to enter the dictionary or propagating operations that may lead to

inconsistencies in the dictionary. Duplicate entities and relationships or recursive generalisations are not allowed, as are duplicate attributes within the same entity/relationship. Two relationships are defined to be the same if they have the same name, member entities and direction. Delete and rename operations are propagated throughout the hyperdocument (dictionary and diagrams) to restore consistency. For instance, deleting an entity leads to the deletion of all relationships in which the entity participates. Modifications to the constraints of a relationship are propagated throughout the hyperdocument.

Selection mechanisms, unlike typing, bypass validation because whatever is offered the designer for selection has been vetted. It ensures that objects of a design space are defined before they are used. Hence, the member entities of a relationship always exist and dependency/relationship arcs always link drawn attributes/entities.

b. Design critiquing.

Critiquing is important to the very nature of database design. Temporary inconsistencies may be allowed to exist as better options are sought. Often these inconsistencies are forgotten, deemed harmless or ignored. It is not uncommon to see dangling relationships, unrelated entities or entities without keys. These conditions, though not 'errors', are undesirable in a database design document on which the development and maintenance of the resulting database is based. The critiquing facility, when invoked, alerts the designer of such conditions. The facility helps the designer to locate the node representing the offending concept (see browsing below) but it is up to the designer to remedy the situation.

Critiquing is done on an entity by entity basis. The designer selects the basis on which to conduct the critique, and the facility does the analysis. There is an additional facility to automatically show the previous or next entity whose semantics may be suspect. Intra- and inter-entity semantic analysis is performed on the selected entity. The tool checks for the following:

i. Entities with no attributes.
ii. Entities with no primary (possibly inherited) keys.
iii. Entities not participating in a relationship.
iv. Duplicated attributes due to inheritance.
v. entities that are possibly synonymous or relationships that seem to be capturing the same semantics.

Figure 4.8 illustrates use of the critiquing facility. The figure illustrates how two entities, 'student' and 'gradStudent', may be synonymous or related in some way not apparent to the designer. Here the facility identifies those relationships with the same name and constraints but involving different entities besides the entity under analysis. It also recognises that in the relationships 'department runs course' and 'course offeredBy department' each relationship is possibly capturing the same semantics. The example illustrates the notification of two homonymous relationships. It is also capable of identifying transitive equivalences such as ['customer orders product'] and ['customer places order', 'order forA product'].

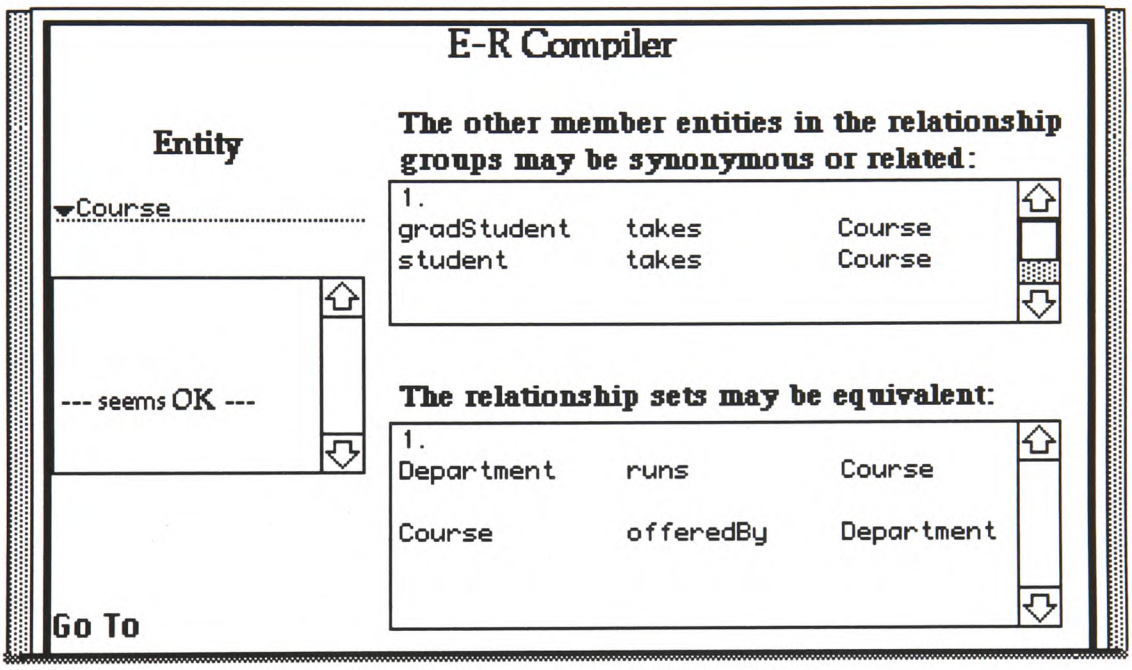

Figure 4.8: A Sample Entity/Relationship Critique for the Entity 'Course'.

5.4. Browsing.

The ability to browse through design documents is unquestionable, more so in an environment that organises them as nodes. Two forms of browsing are envisaged as desirable: local browsing (browsing within a context) and global browsing (browsing across contexts).

a. Local browsing: The major contexts are scrolling fields, backgrounds and stacks. The dictionaries use scrolling fields to represent lists of attributes and data types. These nodes offer local browsing with respect to themselves by virtue of the ability to scroll desired information into the view window.

The designer does not need to exit the node (and indeed the card or background containing the field in question) so the frame of reference of the windowed information does not change. Backgrounds are used as organising contexts for entities, relationships or diagrams. Here the designer needs to move from node to node but still within the same background. HSMS offers 'next', 'previous' and 'find' facilities e.g. the user is taken to the next defined entity. The entity dictionary also offers a hierarchical browser used to traverse the generalisation or aggregation hierarchies. This browser imposes some context-sensitivity on the 'next' facility by actioning an upward traversal. The 'find' facility presents the designer with a list of entities or relationships to choose from, appropriately followed by a traversal to the node containing the attribute, entity, relationship or diagram.

b. Global traversal: these traversals allow traversals across contexts. While viewing an entity, a relationship or a diagram, the user may decide to go to an associated node residing in another context.

We have described the features and facilities of HSMS. Next we compare HSMS with other schema modelling tools.

6. Related Work.

The work discussed in this chapter is related to several other research efforts. [Kambanis 1990], for instance, describes the functionality and architecture of a Semantic Database Modelling Environment, the TDE (Taxis Development Environment). HyperCASE is another effort that is similar to this work [Cybulski and Reed 1992]. Comparisons are also made with a number of CASE tools for the Macintosh described in [Fogel 1992].

TDE is a development environment incorporating tools, techniques and multi-view representations of objects maintained in it [Kambanis 1990]. Both TDE's and HSMS editors support the abstraction mechanisms of aggregation, generalisation, classification and association. However, despite some similarities in consistency and completeness checking, HSMS offers elaborate query and critiquing facilities. TDE dynamically produces an overall E-R diagram which can be optionally hierarchically contextualised around a selected entity (referred to as the focus) as entity editing proceeds. In contrast, HSMS allows for the manual production of linked sub-diagrams (E-R and F-D) and encourages clustering entities by offering selection of the 'focal' entity equivalent. The permanency of HSMS diagrams offers many possibilities,

such as better support in producing E-R diagrams for an information area and appropriately naming and describing them [Feldman and Miller 1986]. Modularising E-R diagrams in HSMS also offers opportunities to restructure the diagrams in communicative ways as well as reducing the cognitive overload, optionally aided by use of multiple windows. Also, unlike HSMS, TDE does not seem to offer manipulation of diagram objects such as entities and relationships. TDE's hypertext view (free-text description, comments) are also offered in HSMS.

HyperCASE, a CAISE toolkit, is an attempt to produce a 'hypertext-based software engineering environment' incorporating a dictionary subsystem (HyperDict), an editing facility (HyperEdit: text and graphics) and a knowledge-based document repository (HyperBASE) [Cybulski and Reed 1992]. HyperBASE offers the following major tools: a document manager, configuration/version manager, project tracker, design tracker and text analyser. It also offers the following auxiliary tools: reuse manager, integrity and completeness manager and design animator. HyperCASE can be considered as an I-CASE system, and as a result it has a wider life-cycle coverage.

[Fogel 1992] gives a survey of data modelling CAISE tools for the Macintosh. Like HSMS, these take advantage of the powerful graphical user-interface inherent in the machine as well as connectivity options offered by the platform. Fogel's work focused on ERDs and examined the tools with respect to forward engineering (the creation of relational tables from schemas). The tools examined are Deft [Sybase], Erwin [Logic Works Inc.], Silverrun [Computer Systems Advisor Inc.], DataModeller [Iconix Software Engineering Inc.] and MacAnalyst/Combo [Excel Software]. Among them Silverrun, Deft and Erwin support forward- and/or backward- engineering, features lacking in HSMS. Some of these tools forward-engineer relational tables (e.g. Silverrun's RDM tool), physical schemas or screen prototypes (MacAnalyst/Combo). Instead HSMS is purely a tool for conceptual modelling along with MacAnalyst/Combo, DataModeller and the ERX facility in Silverrun. Nonetheless, together with the schema integration facility discussed in Chapter 7 they form a toolkit for conceptual modelling, unlike all the tools discussed above. Silverrun offers a tool for DFD-support unlike HSMS which goes the FDD way. A fundamental distinction can be made in that while DFDs focus on data processing, FDDs contribute more positively towards the meaning of data (the domain of conceptual modelling). Erwin and MacAnalyst support the abstraction mechanisms of subtyping, thus

similar to HSMS in this respect. Further, Silverrun supports the 'normalisation' of the produced schema via an expert system. Despite the presence of the functional dependency diagramming facility which would go a long way to support this process, HSMS does not support schema 'normalisation' to discourage the premature creation of artificial entities. Erwin is similar to HSMS in another respect: defining data types for attributes. There is however no error checking in Silverrun, Erwin and DataModeller. Some tools offers multiuser access to their data dictionary (Silverrun, MacAnalyst/Combo and DataModeller). MacAnalyst/Combo further supports O-O analysis and a requirements database unlike HSMS, but a similar way to clustering supports subdiagrams. DataModeller can be considered meta-CAISE in its support for various notations. Unlike HSMS, Erwin and DataModeller offer no error checking, a facility offered in Deft.

HSMS has restricted graphic capabilities. Though not a major part of the work, the performance of HyperCard is worth mentioning and indeed some valuable time was spent in trying to improve it (through the development of faster modules in Pascal). Initially, the major restriction was the limit on the display size of a node even though improvements were offered by HyperCard 2 over HyperCard 1. While such a restriction enforces modularity, the ability to determine the size of a node is invaluable for the data analyst as there is no way to determine in advance the size of an information area or cluster. A secondary limitation was the initial lack of multiple windows in HyperCard 1, which was also alleviated when HyperCard 2 was shipped. Instead of only one HyperCard window, HyperCard 2 offers multiple windows each corresponding to one HyperCard stack. However, restrictions were still encountered in attempting to view, in different windows, two nodes that are resident in the same stack. The alternative, would have been to have a single stack for each entity/relationship/diagram, but it would have further throttled performance and loss of contexts which were felt to be more important.

The work on HSMS based on the given hypermedia system (HyperCard) showed that hypermedia was more useful as a repository (dictionary) than as a diagramming environment. This was partly due to the fact that the node requirements (hence their layout) for a dictionary entry (the templates) can be predetermined unlike diagrams where the layout is dynamic as the diagram is edited. For instance, in a diagram, no two entities may be allowed to overlap for the sake of communicability.

7. Conclusion.

In this chapter we have described the features of HSMS, a prototype tool developed to demonstrate some applications of hypermedia technology to schema modelling. We showed how a hypermedia environment is used to support the process of schema modelling: representing, presenting and managing schema components. We discussed the schema manager, the dictionary subsystem for recording entities and their relationships, the E-R diagrammer and the F-D diagrammer. Also discussed are facilities for browsing, querying, linking and critiquing schemas offered by the prototype. Finally, HSMS was compared and contrasted with a number of related tools.

The main theme emerging from the chapter is that based on the given hypermedia system (HyperCard), hypermedia is more useful for document management than for diagramming purposes. Entities, relationships and attributes can be easily documented as self-contained units and organised into higher-level abstractions (contexts). As a result the chances of getting lost in the network of nodes is reduced as the number of nodes is effectively (logically) reduced. While a facility that helps to cluster diagrams according to functional closures based on selected entities is provided, the process of developing diagrams is relatively lengthy. A corollary to this problem is the imposed diagramming inflexibility resulting from clustering. The performance of the tool is acceptable for small schemas only.

Up to now the discussion has focused mainly on schema modelling. We have discussed the development of a hypermedia schema modelling environment and illustrated how it is used. Now we focus our attention on the next phase of conceptual modelling: schema integration. The discussion will address the problems associated with schema integration and propose collaboration as a possible solution. A working prototype demonstrating the features of a CAISE environment for such collaborative schema integration is then discussed.

Chapter 5

The Process of Schema Integration.

Summary.

The concept of schema integration was introduced in previous chapters. This chapter focuses on the process of schema integration and begins with an introductory section which outlines the principles of the process. We discuss the need for schema integration, and the problems that arise in section 1. Section 2 discusses the various approaches to schema integration. Schema integration can be roughly split into two phases: preintegration (analysis of the schemas and concepts to be integrated and the recording of any pertinent information) and merging (the integration proper). In section 3 we discuss a methodology for integrating concepts. It mainly discusses preintegration: comparative analysis of semantics and a set of encoding constructs to represent its findings (semantic equivalences and semantic conflicts). Section 4 is devoted to merging and the techniques that may be employed in aligning concepts. Section 5 discusses some issues relating to the use of the extended-entity relationship model during schema integration. View integration CAISE is discussed in section 6. Section 7 concludes with the assertion that schema integration is an ill-structured process, discusses computerised support for the process and proposes collaboration as essential for such computerised support.

1. Introduction.

One fundamental requirement for a database is sharability, a factor dependent on the conceptual schema produced during logical modelling (c.f. Chapter 2). Ordinarily, the conceptual schema for a large domain is difficult to produce in one go or indeed by one data analyst. For this reason, many propose that conceptual modelling be split into two stages [Vossen 1990, Elmasri et al 1987]:

a. schema modelling considers the needs of each user (or group of users) separately (as discussed in chapters 2 and 4), resulting in a collection of subschemas.
b. schema integration then articulates and combines the subschemas into a conceptual schema that satisfies the requirements of all users.

The assumption for (a) is that conducting a data analysis on a small data space is easier than on a large one. This is because the resultant schema is more likely to be easier to comprehend. Several issues relating to subschemas have

to be addressed so as to retain their semantics, remove redundancies and eliminate conflicts in the conceptual schema. Schema integration makes use of knowledge from various sources. At the model level, the knowledge can be found embedded in the descriptive attributes of the entities, relationships or attributes in question. Additional knowledge can be found in the information areas themselves as well as integrator expertise. Figure 5.1 shows a schematic representation of the aspects that may lead to ambiguities.

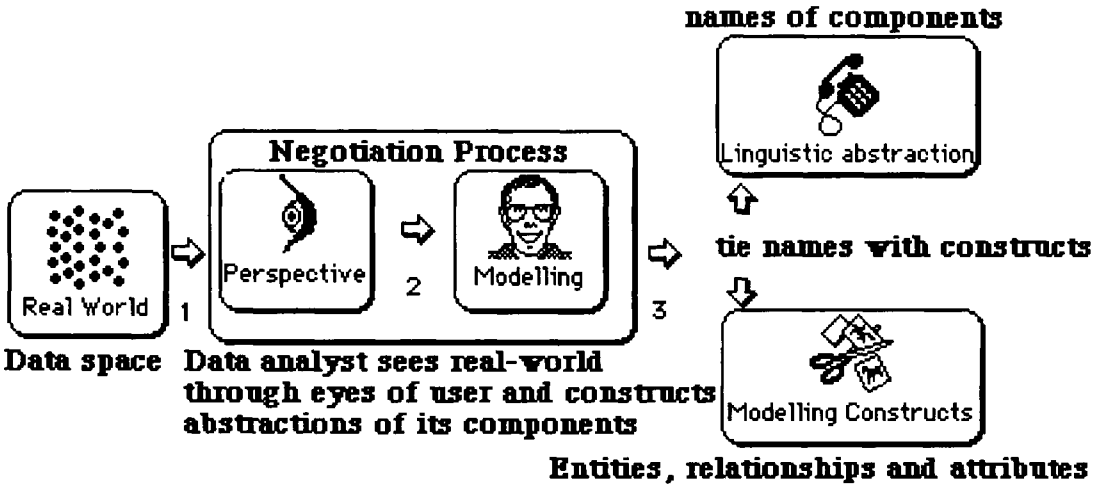

Figure 5.1: Schematic Representation of Conceptual Modelling Process.

Schema integration may be influenced by the permanency of the integration, the number of subschemas, the independence of databases and the sources of the subschemas (externally generated subschemas may be difficult to integrate). We assume that schema integration is deemed necessary. For subschemas requiring integration, the integration is necessary because:

a. Schemas may represent the same information area but were produced from different perspectives of the information area. The respective observers (users or data analysts) of the information area do not perceive the same set of properties in the information area. Often this is a consequence of the many forms of information available in the 'real-world' for the information area under consideration. For example, two perspectives of a student may be defined as follows: Student (FirstName, Surname, NINum, Age), and Student (name, SSN, DateOfBirth, Sex). Obviously, these are only subsets of the information about a student that an academic institution holds, for instance the information about a students country of origin is not captured in the above schemas.

b. Overlapping schemas: schemas may represent different but related information areas. This is a direct consequence of the 'divide-and-conquer' approach to conceptual modelling. Some information areas may share common data, hence a design based on information areas may duplicate some data which may in fact represent some interschema relationship [Batini et al 1986]. Extending the previous example, consider the relationships: (Student enrolledIn Department) and (Student occupies HostelRoom). Assuming that the student entities in the two relationships refer to the same real-world concept, the two schemas are related via the common entity student.
c. Incompatible modelling constructs: schemas may be modelled using different data models e.g. the E-R model, the relational and the OO models (c.f. Chapter 2, section 3). Such schemas are not always easy to integrate as a canonical data model has to be found for the integrated conceptual schema to avoid the possible loss of important modelling information [Sjoberg 1993]. Further, not many designers have expertise in several otherwise conflicting data models. This work assumes that the schemas to be integrated are based on the same data model, the Extended E-R model (c.f. chapter 2, section 2), so there are no data model incompatibilities. For instance in the relationship (Student occupies HostelRoom), a small college might not attach much significance to residential services so it stores a student's hall information as an attribute of the student entity (FirstName, Surname, NINum, Age, ..., HostelRoom, ...). This amounts to incompatible modelling constructs (an entity and an attribute) representing the same information (about hall allocation).

The problem is summarised in Figure 5.2 where R represents a 'real-world' object with two perspectives P1 and P2, represented by two abstractions A1 and A2. The abstractions branch into two linguistic abstractions L1 and L2, and two modelling constructs M1 and M2. The requirement of schema integration is that given either the linguistic abstractions (L1, L2) or modelling constructs (M1, M2), we can backtrack along the two chains to get back to the same real-world object R which we then represent with only one abstraction. The representation of reality R is in effect filtered through stages 1 to 3, particularly stages 2 and 3, thus complicating the schema integration process.

In a special SIGMOD issue on heterogeneous databases, [Sheth 1991] identifies two basic semantic issues relevant to schema integration: determining how concepts between subschemas are related (semantic

equivalence and semantic conflict), and how such concepts can be reconciled (semantic reconciliation).

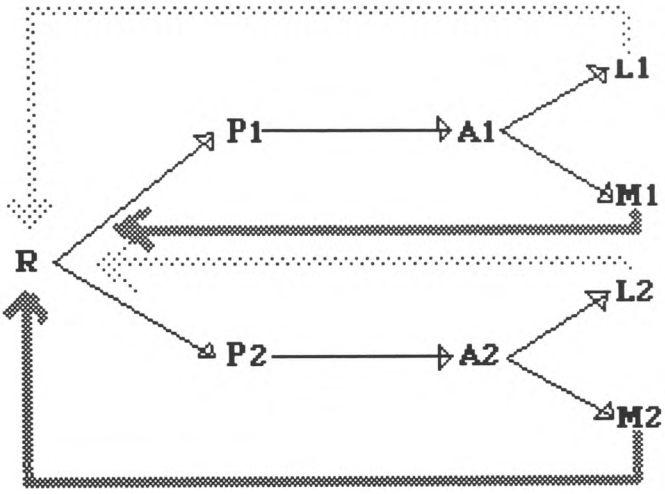

Figure 5.2: Equivalence Establishment Requirements for Integrability.

A semantic equivalence exists if subschema components refer to the same real-world concept. Semantic conflicts exists when subschema components representing the same real-world concept have different information or are different. Semantic equivalence and conflict are complementary since one sheds light on the other e.g. if two concepts are semantically equivalent and they are not represented the same in their respective subschemas then their representations are in semantic conflict. For purposes of this work, the term semantics refers to structure, integrity constraints and operational rules of the components in EERM-based subschemas. The terms equivalence and conflict will be used to denote semantic equivalence and semantic conflict respectively.

a. Semantic Equivalences.

Different user perspectives of data lead to the capture of different semantics of the data. For instance, one user may not be interested in some attributes of an entity while another is (e.g. sex in Figure 5.3). A consequence of differences in interest include differences in representation detail and of derivable concepts. For the given illustration, 'name' may be computed as a concatenation of 'FirstName' and 'SecondName', and similarly 'age' may be computed from 'DoBirth'. Other forms of equivalence may exist, such as differences in units (e.g. fees per year and fees per term), ordering-format in multivalued attributes (e.g. date as DD/MM/YY or MM/DD/YY) and underlying data type mismatches (e.g. 'character' and 'integer' for studentID).

b. Semantic conflicts: modelling construct incompatibilities.

Differences in linguistic abstractions (names) mainly appear in the form of synonyms (e.g. NationalInsuranceNumber and SocialSecurityNumber) (c.f. Figure 5.3) and homonyms (e.g. faculty as in department and faculty as in human ability). Such differences are often referred to as conflicts for, by definition, a named concept must be uniquely identifiable as such.

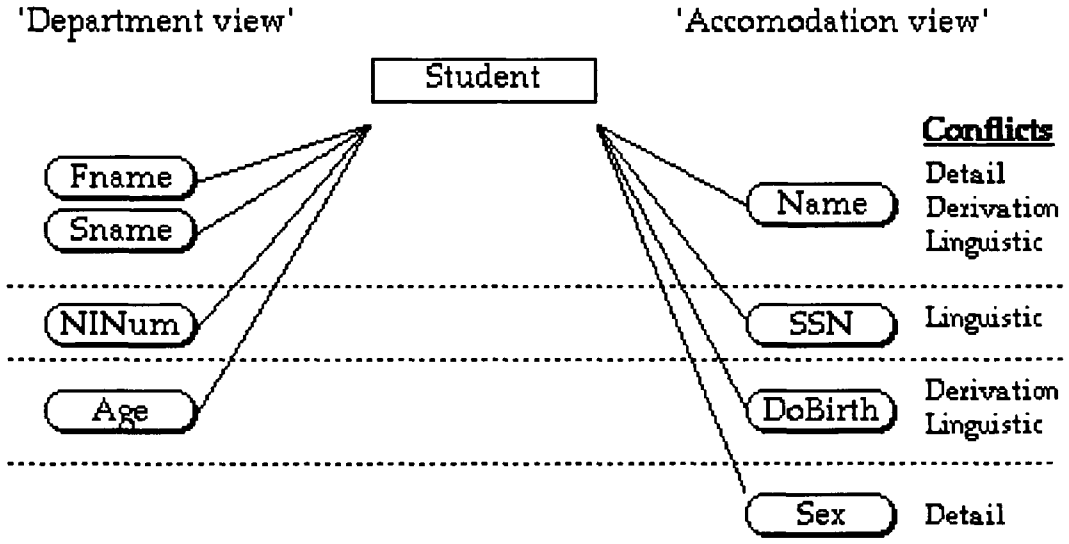

Figure 5.3: Interschema Conflicts About the Entity 'Student'.

A concept might be an entity in one schema and an attribute in another e.g. the concept publisher might be an entity of a library schema and an attribute of the entity book in a publishing-house schema. Man and woman are specialisation entities of the entity person, but the value of the attribute sex in the person entity might be used to differentiate the two entities. These constructs must be viewed as capturing the same semantics despite different structural representations.

However, articulating these constructs requires a clear understanding of their meaning, constraints and the consequences of any changes effected on them. We define these terms again with a view to highlighting the semantics implicitly captured by their use.

i. Attribute.

An attribute is a property of an entity or relationship. The underlying fact about an attribute is that it belongs to concepts of a particular type i.e. it models the relationship between a lower level concept (the property) and a higher level concept (the concept with that property: an entity for instance).

An attribute, therefore, can only participate in a relationship indirectly through the participation of its entity in the relationship. More importantly (and implicit in the above definition) is the fact that an attribute takes-on values from a well-defined domain which defines its structure and some behaviour [Leonard 1992]. The implication of this notion is that in order to merge two attributes, they must share a unique domain (c.f. section 3.1(a) below), and the resultant attribute must have that single domain. Hence to merge two attributes that have totally unrelated domains violates the definition of an attribute.

ii. Entity.

An entity is a concept about which we want to record information (properties) in a database [Date 1990]. Concepts of the same entity type are expected to have the same structure and behave homogeneously. The structure is represented by the totality of the attributes composing the entity. The behaviour is based on the totality of the behaviour of its attributes (internal data dependencies) and the operations that can be performed on it.

iii. Relationship.

Entities have relationships amongst themselves that represent organisational rules. Concepts of the same relationship type are expected to have the same structure and to suffer the same transformation under the same operation. The structure is represented by the member entities and the attributes of the relationship itself as well as the constraints (cardinality and optionality) defined upon it. The behaviour is based on the totality of the dependency constraints among its attributes and the member entities.

The data integrators decide on the best form of representation of the concepts by taking into consideration knowledge embedded within such constructs (e.g. abstraction level, structure, behaviour) and the associated loss/gain in semantic representation.

2. Approaches To Schema Integration.

Approaches to schema integration can be categorised in terms of the number of concepts integrated at a time and the abstraction gradient (top-down or bottom-up) the concept integration process follows at each stage.

a. Strategy on number of concepts to be integrated.

The concepts to be integrated are selected from a set of available concepts (including the current evolved concepts), thus data integrators have to decide on the number of concepts to integrate at a time. Figure 5.4 shows several techniques that have been suggested for this [Batini et al 1986]. Given **n** concepts, data integrators can:

i. integrate all **n** concepts at once (a **one-shot n-ary strategy**), or

ii. integrated a subset of the **n** concepts repeatedly until all are done (**iterative n-ary strategy**).

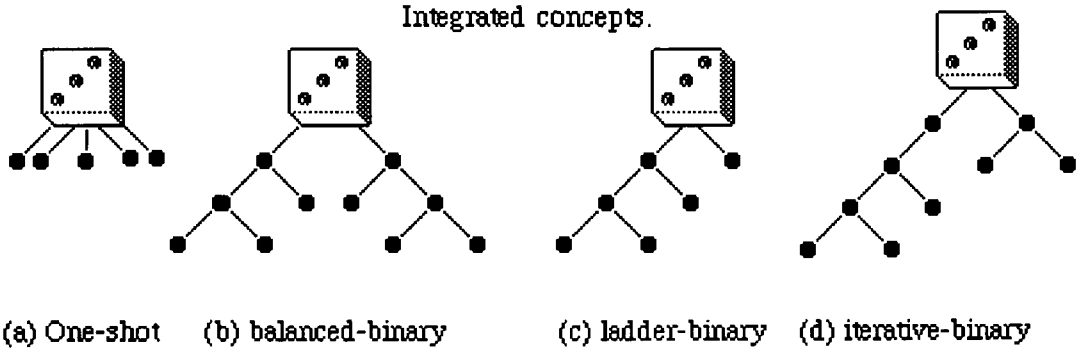

Figure 5.4: Concept Integration Strategies.

The iterative strategy seems to be more attractive as it is flexible and it does not overwhelm data integrators with concept knowledge for integration. The complexity of the comparison and integration stages is reduced if n is small. The simplest strategy is for n = 2 (a binary strategy: only two concepts are being integrated) of which three variants are recognised [Batini et al 1986]. If concepts are paired-up such that the integration process is symmetrical then the integration process is termed a **balanced binary strategy** (Figure 5.4 (b)) and it has been claimed to converge faster. If a concept is integrated with an evolved concept at each step then the strategy is a **ladder binary strategy** (Figure 5.4 (c)). The selection of concepts may be based on weights attached to input concepts as proposed by [Batini and Lenzerini 1984] or by presupposing that an evolved concept has the greatest weight [Batini et al 1986]. The former approach may not be the ideal solution in general since the relative weights among concepts may vary after each major integration step. Preassigning weights to concepts only appears natural to concepts that conform to a strict hierarchy. An **iterative binary strategy** (Figure 5.4(d)) selectively chooses concepts at each stage with no regard to preassigned weighting: this avoids having to deal with concept-weight variations.

b. Abstraction gradient: top-down or bottom-up concept integration?

Each integration process produces a concept at least at the same level as one of the input concepts. This requires that the level of abstraction of a concept may have to be raised or lowered (conforming) to ensure integrability with a concept at a different level (c.f. Figures 5.5 and 5.7). There are three basic constructs in the ER model, namely: **attribute, entity** and **relationship**, in increasing order of abstraction and complexity. For instance an entity is composed of attributes, and a relationship is in turn composed of attributes and entities. Figure 5.5 shows a simple diagram of how these constructs may interact during schema integration. The presence of more than one basic structure in the ER model and its extensions complicates the formation of mental mapping models as data analysts have to transform from one structure to another for comparison purposes [Saltor et al 1991]. The basic integrations can occur between:

i. two entities,
ii. an entity and a relationship, and
iii. two relationships, and these are based on
iv. two attributes,
v. an attribute and an entity.

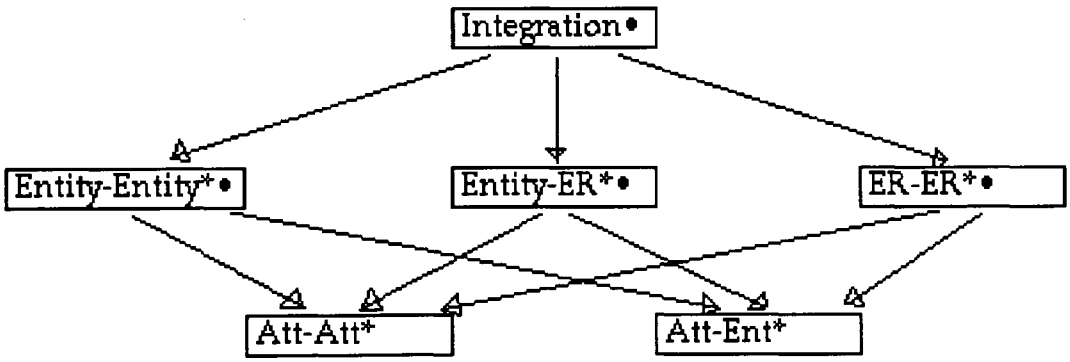

Figure 5.5: Concept Integration Paths Through Abstraction Levels.

Data integrators are therefore faced with taking either a top-down or bottom-up approach to concept analysis (the middle-out approach is by definition excluded because we already know the boundaries of one of the input concepts and what has to be conformed). The top-down approach results in a less abstract concept while the bottom-up approach results in a more

abstract concept. In doing so, a data integrator creates a conformed picture of a transform of one of the concepts that is integrable with the other. For example, the idea behind converting an attribute to an entity is to make the resultant entity integrable with a specific entity.

From the user's point of view a lower abstraction is more appropriate as it is specific, hence more comprehensible. On the other hand, a higher abstraction is more appropriate from the data analysis point of view as it is more general, hence more flexible and comprehensible because of information hiding. It might, however, compromise comprehensibility to the users who are generally not conversant with abstraction techniques (in particular aggregation and generalisation).

From the foregone discussion it can be concluded that:

i. because of the structural notion in ER abstraction, the least requirement for the integrability of entities and relationships is the integrability of some of their attributes. Higher abstractions are integrable if and only if some of their lower abstractions are. In particular, attribute-based integration guarantees that one basic structure is used as the basis for integrability.

ii. the bottom-up approach is more pragmatic as it ties naturally with (i) above, and in the process produces more stable and general abstractions. That is, with a bottom up approach equivalence knowledge is based on specifics which the users can directly relate to (such as actual data items (the attributes)), then as the equivalences are verified, more abstract structures are built that data and systems analysts are more comfortable with. Further, this seems a more pragmatic approach when dealing with abstraction hierarchies (as discussed later in section 5) as abstraction hierarchies can be recursively synthesised.

3. Methodology For Concept Integration.

[Batini et al 1986] gives a comprehensive list of concept integration methodologies fundamentally comprised of the following phases: preintegration, comparison, conforming, merging and restructuring. Depending on the philosophy and the data model employed, some methodologies utilise all these phases [Elmasri et al 1987] while others do not [Motro 1987]. For example, integrations based on the functional data model do not need a restructuring phase as attribute is the only data construct utilised. In this work we define concept integration to be phased as follows: preintegration and merging. Preintegration establishes and

gathers facts about the integrability of concepts, while merging does the actual integration. Preintegration comprises concept selection, equivalence and conflict analysis. Figure 5.6 shows these stages. The figure shows conflict analysis preceded by equivalence analysis following [Larson et al 1989]'s postulate that conflicts are deducible from equivalences as follows:

> If two concepts are semantically equivalent (in the real-world sense) and they have different modelling constructs, then a conflict exists among the modelling constructs. In the same vein, if two concepts are not semantically equivalent and they have the same modelling constructs, a conflict also exists among the modelling constructs.

Equivalences and conflicts are stored in an integration data dictionary, and are used in the merging phase to produce integrated concepts which are also stored in a concept dictionary.

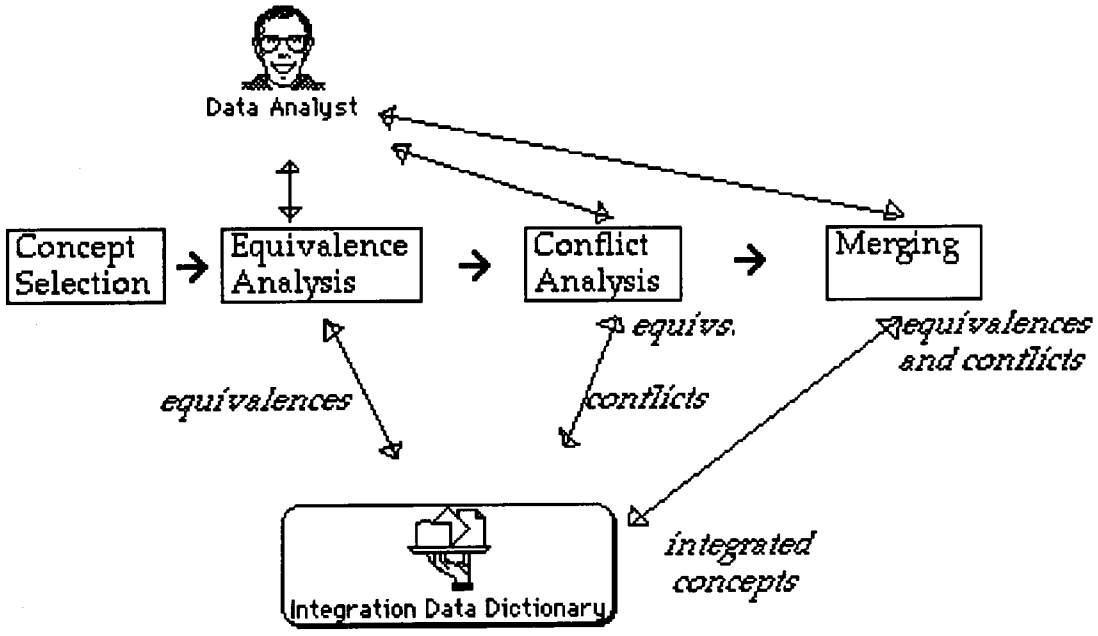

Figure 5.6: Phases of Concept Integration.

3.1. Preintegration.

Preintegration answers the question: 'given a representation M1 for a real-world concept R in subschema A, is there in subschema B:

a. a representation M2 of R (if so, M1 and M2 are equivalent), or
b. a representation M1 of a different real-world object R2 (if so, M1 is at the centre of a conflict of representation of R1 and R2)?'.

Several issues interplay to answer this question, as such incompatibilities may be in the represented semantics (and their relative depths) as well as in mismatches between the modelling constructs (attributes, entities, relationships) used to encode them (c.f. section 2).

Preintegration consists of the following aspects: concept selection, equivalence and conflict analysis, and their specification.

3.1.1. Concept selection.

The concepts to be integrated are selected from the pool of available concepts (including the current evolved concepts) as discussed in section 2(a) above.

3.1.2. Equivalence and conflict analysis.

Schema integration is basically about identifying equivalences and conflicts, and then eliminating them from the integrated conceptual schema. A distinction can be made in that equivalences are about the real-world data and conflicts are about the abstraction constructs (e.g. names) we create for the data in our models. The data integrators examine the selected concepts with an eye to identifying and specifying the equivalences (in the domain) and conflicts between them (in subschemas).

a. Equivalence identification.

The problem of concept integration has been widely discussed in the literature [Jajodia et al 1983, Larson et al 1989]. Two concepts are considered equivalent if they are:

i. domain equivalent: there exists an isomorphic function from the domain of one concept (known as the domain of the function) to the domain of the other (known as the range of the function) [Larson et al 1989]. This ensures that the range instance is an alternative representation of some domain instance.

ii. semantic equivalent: the two concepts have equivalent internal semantic properties such as integrity constraints (both static and dynamic) [Larson et al 1989, Jajodia et al 1983]. This ensures the integrity constraints hold for both concepts.

iii. operational equivalent: the set of database operations performed on one concept has an equivalent set of operations in the other [Larson et al 1989]. The basic database operations are those that alter the state of a database,

namely create, insert, delete and replace. This allows data integrators to compare the behaviour of the various instances under external stimuli.

E-R model based schema integration has been widely discussed in the literature [Navathe et al 1986, Larson et al 1989, Jajodia et al 1983]. In particular, the work of [Larson et al 1989] established a sound theoretical basis for attribute equivalence and extended it to entity and/or relationship equivalence. They defined a number of properties for attribute equivalence, the most important of which is the existence of an isomorphic (invertible) mapping between values of one attribute and those of another. This formalism relies on a good understanding of the semantics of the input schemas as well as the completeness and accuracy of the mapping models generated by the data analyst.

This section borrows from the theory of attribute equivalences postulated by [Larson et al 1989]. Two attributes are defined to be equivalent if one is an alternative representation of the other. Equivalence among attributes may hold momentarily or for all time. Equivalences that hold for all time guarantee integrity for all instances of the integrated attribute (while the former do not). Without loss of generality, all discussion on equivalence hereafter refers to those that hold for all time.

Let x and y be two concepts selected for integration and let Dom(x) be the domain of x (set of values of x **for a particular perspective** of x). A concept may technically have alternative domains, each dependent on the perspective, on condition that all of the alternative domains are subsets of the same containing domain, possibly composed of the sub-domains. The process of semantic analysis aims at defining the mapping amongst these domains.

The preceding discussion defined the term domain, but how does this concept assist in equivalence identification?. Firstly, the usefulness of a domain is that the ability to share domains is an indication that two concepts may be integrable. Secondly, the domains also guarantee that meaningful assignments and comparisons are prevented in much the same way as in strongly-typed programming languages. For instance, both the 'age' and 'year-of-study' of a student have the same underlying physical data type (integer: which is too broad to form a basis for a domain), yet merging them is both pointless and may be dangerous because the notion of such physical data structures is too broad.

b. Equivalence Specification.

Equivalences can be formally defined by mapping assertions between domains (set of values) of concepts. The concept of domains has influenced many works to narrow down the problem of semantic equivalence/conflict to domain equivalence/conflict [Navathe et al 1986, Larson et al 1989, Hayne and Ram 90]. Defining mappings amongst the individual elements of the domains is undesirable for a number of reasons. Firstly, the number of values in any domain can be very large. Secondly, the data analyst is not interested in the values, but the answer to the question 'Given two concepts, do (or can) they share the same domain?'. It suffices to define the mapping abstractly and concisely using formal models so that integration ideas are abstractly verifiable, communicable and open to criticism [Olerup 1991]. Following the works of [Sheth et al 1988] and [Navathe et al 1986], a mapping can be defined in terms of the following equivalence types among the respective domains:

i. Equal equivalence: there is a 1-1 correspondence between all elements of Dom(x) and all elements of Dom(y).
ii. Contains equivalence: there is a 1-1 correspondence between a subset of Dom(x) and Dom(y).
iii. Overlaps equivalence: there is a 1-1 correspondence between a subset of Dom(x) and a subset of Dom(y).
v. Disjoint equivalence: Dom(x) and Dom(y) are disjoint, same-typed and coexist.
iv. Incompatible equivalence: there is no correspondence between Dom(x) and Dom(y).

An equivalence falling into category (ii) may be formally stated as (concept-1, Equivalence Type, concept-2) e.g. (Student, Contains, underGraduate). Such a formal statement captures an integrate plan, and it consists of 'which' concepts to integrate or disambiguate ('student' and 'undergraduate') and 'why' they are integrable ('contains'). The equivalence statements are submitted to the integration data dictionary and are later made use of in providing the semantics for merging and communicating integration plans.

c. Conflict identification.

Following [Larson et al 1989], if two concepts are semantically equivalent (in the real-world sense) and they have different models, then a conflict exists among the subschemas. The conflict may manifest itself in the form of the

modelling constructs themselves (i.e. abstraction levels) or the names associated with the same modelling constructs of the same real-world concept. We outline below how this may be achieved for naming conflicts, but the principle is the same for modelling constructs. Deduction of naming conflicts (synonyms and homonyms) based on verified equivalences can be achieved as follows:

i. **Synonyms**: If two concepts are verified equivalent, their names are compared. If they are different, then the two names are synonymous.
ii. **Homonyms**: If two concepts at the same abstraction level have the same name and they are not verified as equivalent, then they are homonyms. In such cases the data analysts have to rename one of the concepts.

This notion of naming discrepancies is sound since synonyms and homonyms have deeper meaning for entities and relationships. Consideration has to take account of not only the assigned names, but also whether the keys of the involved entities or relationships are the same (i.e. homonyms) or different (i.e. synonyms).

d. Conflict specification.

Adopting the strategy for conflict identification discussed in (c) above, conflict specification and storage in the dictionary is unnecessary as these can be computed from the formally stated and verified equivalences already stored in the dictionary. However, some approaches may require that these conflicts are specifically stated. [Storey and Goldstein 1990] have proposed the use of synonym/homonym lexicons (a list pairing-up synonyms and homonyms) stored in the integration data dictionary.

4. Merging.

Having all the equivalence and conflict assertions, data integrators proceed with the actual integration. They superimpose the concepts into an intermediate integrated concept. Merging requires that a specification of the resultant concept be given and this normally entails applying a transformation to one or both of the input concepts to construct the final concept. Mergers can be horizontal (i.e. same level of abstraction) or vertical (across abstraction levels). For both, the data integrator specifies the concept that will finally appear in the conceptual schema, thus implicitly states the desired concept transformation (see Figure 5.7).

As for preintegration, transformations may be formally stated as (concept-1, concept-2, how-merge, new-concept) e.g. (Student, underGraduate, Generalise, Student). Such a formal statement captures a transformation plan consisting of 'how' to integrate ('Generalise') and 'what' they integrate into ('Student'). A merger intention can be defined as one of the following operators:

a. 'Union': the merge operator amalgamates the two concepts into one and eliminates the other. For instance the entity foreignStudent may be absorbed into the entity student. However, if there are no key equivalences between the entities the key of foreignStudent has to be prime with respect to the resultant entity.
b. 'Generalise': the 'generalise' operator creates a generalisation relationship between 'contains'-equivalenced entities where there exists instances of the containing entity type that are not instances of the contained entity type e.g. ('student' generalises 'partTimeStudent').
c. 'Aggregate': the 'aggregate' operator creates an aggregation relationship between 'contains' equivalenced entities e.g. (Department isPartOf Faculty).
d. 'Associate': the 'associate' operator creates a named association relationship between the involved entities. This operator is normally used to represent operationally meaningful interschema relationships that were not specified during schema modelling.
e. 'Meet': the term 'meet' operator is borrowed from the work of [Motro 1987]. The operator creates an entity that generalises both entities in an overlaps/contains equivalence e.g. (Student Generalises UnderGraduate) and (Student Generalises PostGraduate).

Similarly, the merge plans are submitted to the integration data dictionary, thus constituting an integration knowledge-base that forms the basis for the penultimate integration as well as communicating integration plans to other stakeholders for agreement.

Figure 5.7 gives an overview of transformation and merging. Transformation may occur between any two constructs except between an attribute and a relationship for the following reason. An attribute by definition is atomic hence it cannot be split to form a relationship which is non-atomic, and similarly a relationship cannot be coerced into an attribute. The type of transformation is determined by the type of equivalence and several other factors such as the effect on other parts of the subschemas (e.g. constraints) and complexity (e.g. partitioning for semantics allocation, understandability).

Usually an equal equivalence leads to a representation that mutually excludes one of the concepts. However, contains equivalence lead to a generalisation/aggregation interschema relationship, while an overlaps equivalence leads to an association, a generalisation or an aggregation relationship as shown in Figure 5.8.

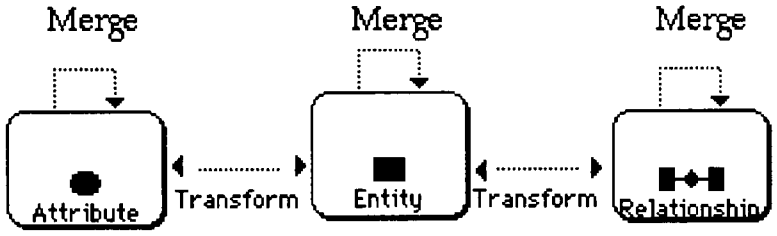

Figure 5.7: Transformation and Merging of ER Constructs for Integration Purposes.

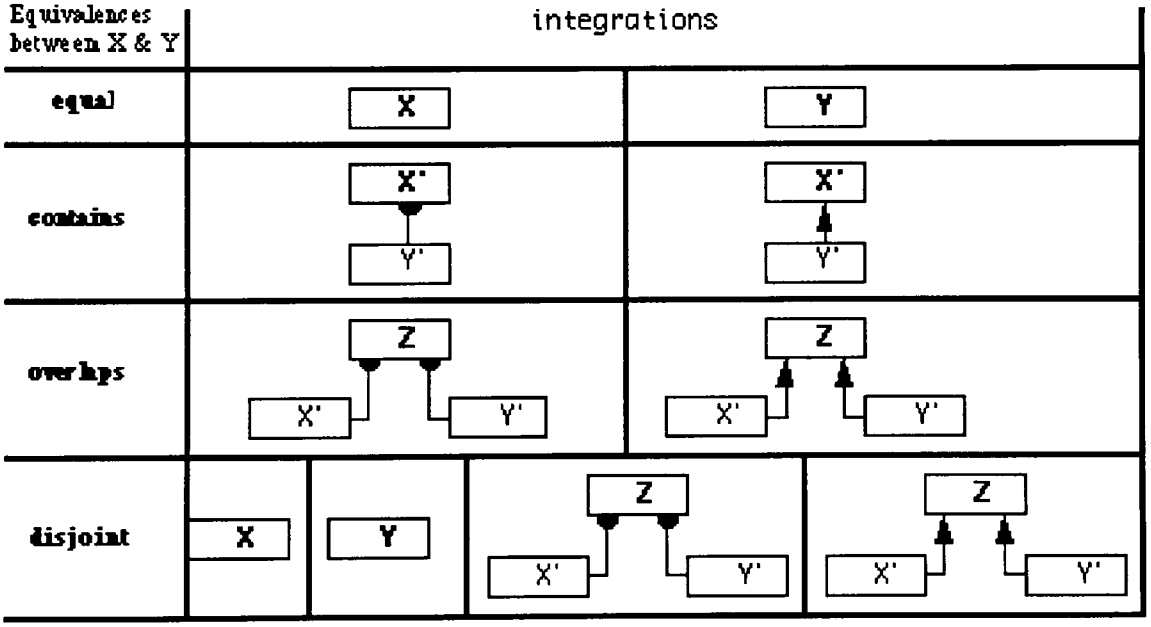

Figure 5.8: Possible Entity Integrations for Given Equivalences.

[Larson et al 1989] lists 4 integration strategies for attributes, namely:

a. integration of all non-disjoint attributes.
b. integration of attributes that have an equal equivalence.
c. integration of attributes that are non-disjoint and indicate the relationship between non-integrated similar attributes.
d. integrate all non-disjoint attributes and migrate values between attributes.

They note that (a) results in the greatest reduction in the number of attributes, followed by (d) and then (b) and (c). These strategies are equally applicable to entities and relationships. [Navathe et al 1986] gives details of integrations for relationships of various equivalences and considers additional semantics such as roles, cardinality and optionality constraints. [Larson et al 1989] also indicates how strategies (a) - (d) above may be adapted to relationship integration by first migrating the relationship and its attributes to both member entities of the equivalent relationships, integrating the member-entity pairs and then establishing a relationship between the resultant entities.

The following algorithm gives an outline for the integration proper:

a. IF Transform (Attribute-i, ->,Entity-j) THEN *(e.g. book.publisher, publisher)*
   Conform (Attribute-i, Entity-k)
   Relate (Entity-j, Parent (Attribute_i))
   Delete(Attribute-i)
   Merge (Entity-k,Entity-j) {go to f}
   END IF

b. IF Transform (Attribute-i, <-, Entity-j) THEN *(e.g. book.publisher, publisher)*
   FOR ALL Attributes A WHERE Attribute (A, Entity-j) DO
   Merge (Attribute-?, Attribute-A) {go to e}
   END FOR
   END IF

c. IF Transform (Entity-i, ->,Relationship-j) THEN *{e.g. Car(..,door,..), Car hasPart Door}*
   Conform(Entity-i, Relationship-k)
   Delete(Entity-i)
   Merge(Relationship-j, Relationship-k) {go to g}
   END IF

d. IF Transform (Entity-i, <-, Relationship-j) THEN *{e.g. Car(..,door,..), Car hasPart Door}*
   Conform(Relationship-j, Entity-k)
   Delete(Relationship-j)
   Relate (Entity-i, SourceEntity(Relationship-j)
   Relate (Entity-i, DestinationEntity(Relationship-j)
   Merge (Entity-i, Entity-k) {go to f}
   END IF

e. IF Merge (Attribute-i, Attribute-j) THEN *{e.g. student.SSN, student.NINumber}*
   Unify (AttributeInfo-i, AttributeInfo-j)
   Delete(Attribute-j)
   END IF

f. IF Merge (Entity-i, Entity-j) THEN *{e.g. underGraduate, postGraduate}*
Unify (EntityInfo-i, EntityInfo-j)
Delete(Entity-j)
END IF

g. IF Merge (Relationship-i, Relationship-j) THEN *{e.g. student takes course, student in course}*
Unify (RelationshipInfo-i, RelationshipInfo-j)
Delete(Relationship-j)
END IF

This algorithm gives a simplified representation of transformation and merging issues. The algorithm can be considered to form a basis for all forms of integration that may emerge by iteratively applying the algorithm sections (a) - (g). A case in point are integrations involving relationships. Thinking in terms of entities, ERs involve at least three concepts (at least two member entities and the relationship itself), which is more than required for a binary relationship. For instance, it is not uncommon to consider a mapping involving the whole ER (i.e. relationship coupled with its member entities) e.g. 'Customer Orders Product' may map to the entity Order (OrderNo, Date, ..., *OrderInfo, CustomerInfo, ProductInfo*). Using the algorithmic sections (a) - (g) as a basis for integration, this integration may be achieved through integrating the concepts 'Orders', 'Customer' and 'Product' in turn into the entity 'Order' (OrderNo, Date, ...).

Details of the actual transformations and mergers have been widely discussed elsewhere [Batini et al 1983, Navathe et al 1986, Larson et al 1989]. Between them, these comprehensive works articulate various forms of equivalence and conflict, culminating in the use of abstraction hierarchies to model interschema relationships (via entities) as shown in Figure 5.8. The concepts at the arrow/blob end aggregates/generalises the other.

5. Schema Integration and the Extended Entity-Relationship Model (EERM).

Figure 5.8 illustrates the power of the abstraction mechanisms of aggregation and generalisation of the EERM in schema integration. However, no consideration has been given to integration of concepts involving these mechanisms in the literature. With the growing prominence of the OO paradigm which heavily relies on these constructs, we cannot afford to continue to ignore them.

The research has usefully highlighted a number of questions that demand future work. The following issues need to be addressed:

a. the implications of recursive structures such as inheritance in generalisation.
b. the implications of possible transformations involving generalising entities. What happens if we transform a generalising entity into a relationship?. Do we lose the various specialisations of the entity, or do we generalise the generalised entity by one of the member entities of the resultant relationship?.
c. the knowledge requirements for aggregation and generalisation hierarchies. Obviously, generalisations and aggregations lead to very abstract models, which provide limited semantics or knowledge for integration on a one-one basis at the same level. Consider the simple example given in Figure 5.9. The knowledge requirements for (b) are much more than that for (a) as most of the detail is hidden away from those integrators who might not know of the existence of the concepts hidden in the recursive hierarchy.

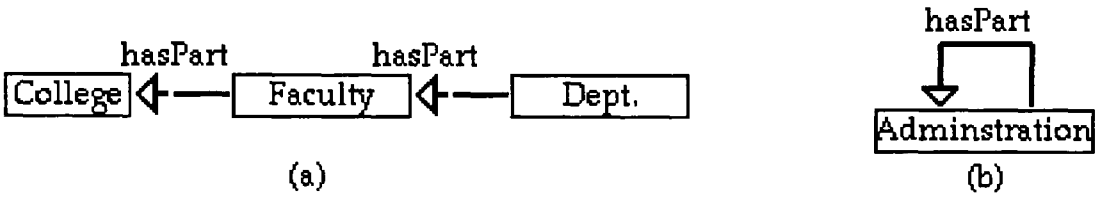

Figure 5.9: Illustration of the Use of Abstraction in Schema Integration.

d. What about cross-level integrations?. In general we expect the overall knowledge requirements to be enormous. For example, an integration of the least abstract entities in abstraction hierarchies has to continually consider all the properties (static and dynamic) defined up the hierarchy. It is however, difficult to know in advance the consequences of an integration.

The underlying problem is that abstraction hierarchies may lead to oversimplification of the 'real-world' concepts, causing the utility and reliability of the concepts and integrations defined upon them to be compromised. Such encoding imply that consideration has to be hierarchy-wide. From this discussion, we can therefore conclude that the abstraction mechanisms of aggregation and generalisation may be more appropriate to

concepts that are not expected to be involved in future such integrations. It seems, they are ideal during the restructuring stage when the conceptual schema is modified for presentability and communication.

6. Schema Integration and CAISE.

Schema integration has not escaped the attention of CAISE [Hayne and Ram 1990, Sheth et al 1988]. This section briefly discusses CAISE for schema integration. A schema integration CAISE tool can offer two features: investigation and integration support.

6.1. Investigation.

The CAISE tool assists in finding and/or managing any assertions between schemas during preintegration. [Sheth et al 1988] propose CAISE support to enhance preintegration in the form of:

a. 'syntactic processing enhancements' which utilise string-matching heuristics leading to the establishment of a synonym, homonym or antonym lexicon [Storey and Goldstein 1990] representing naming equivalences/conflicts.
b. 'semantic processing enhancements' which utilise semantic heuristics leading to the identification of equivalences e.g. common attributes between entities/relationships indicate some form of equivalence. However, a problem still remains in establishing attribute equivalence on which such correspondences are based.

Such a tool may be automated or manual as discussed in the next paragraphs.

An automatic approach is depicted by the MUVIS system which utilises an expert system that utilises a rule-base to compare concepts in a binary fashion, and heuristically computes assertions about the equivalences or conflicts [Hayne and Ram 1990]. Inevitably, this approach still requires that the integrator interactively confirms or rejects the computed assertions. The MUVIS system has gone further in this respect by supplying a computed weighted probability of similarity based on name, key, attribute, relationship and transaction equivalences e.g. equality of names is taken to imply a probability of similarity of 0.5.

On the other hand, a manual (non-automated) approach depends on a knowledgeable integrator forwarding the equivalence/conflict assertion to the system. This is the basis of the approach taken by the work of [Sheth et al 1988] where, through a series of forms, the tool interactively prompts the integrator

for information that is used to construct an assertion. Nonetheless, the supplied assertion has to be checked for consistency and contradiction, an area that can be automated. The philosophy behind this approach is that equivalence/conflict identification cannot be totally automated as data models are by definition negotiated abstractions lacking the detail necessary to firmly establish the equivalence or conflict.

Invariably, the various integration approaches utilise some form of integration data dictionary (a CAISE tool in its own right) to store and manage equivalences and conflicts. For instance, [Sheth et al 1988] utilise what they call an Attribute Class Similarity matrix, Object Class Similarity matrix and Entity Assertion matrix. Such representations are formal structures, thus assertion communication (between integrators, users and the merging facility discussed below) is via well defined protocols.

6.2. Merging.

The tool appropriately merges the involved concepts, and may even create interschema mappings (between subschemas and the conceptual schema) based on given equivalence assertions (gleaned by the investigator, or forwarded by the data integrators: as stated above). The first step is to conform the concepts, for example MUVIS converts between [Hammer and McLeod 1981]'s Semantic Data Model and the EERM. This often involves renaming and enriching concepts by transferring attributes between entities/relationships. The second step is to integrate the concepts (including conformed ones) and this often involves deleting, excluding or linking one of the equivalent concepts in an integrated schema. The former approach is employed for a total integration (i.e. an integration in which the input schemas are completely discarded), as opposed to partial integration (in which only a schema containing resolved concepts is represented). Partial integration suits the design process best as iteration and backtracking can be achieved flexibly. This can be done through judiciously linking and unlinking concepts in the partial schemas and those in the input schemas from which they are derived. A tool to support such linking mechanisms is therefore vital.

7. Conclusion.

In this chapter we have discussed some of the salient features of schema integration. We have noted that concept integration involves stating 'which' concepts to disambiguate, 'why' there is an ambiguity (equivalence and conflict), 'how' it is resolved and 'what' is finally implemented. We have also

discussed the need of CAISE support for managing schema integration. However, it is difficult to know in advance the consequences of an integration. Only through consultation and negotiation with stakeholders can such consequences come to light or their effects be reduced or recognised. In particular, representing the 'why' and 'how' factors is important as an integration (domain specific) knowledge-base can be achieved. For instance, the effect on other parts of a subschema (especially for abstraction hierarchies) needs to be accounted for and propagation of necessary change must be ensured [Sjoberg 1993], and such accounting is broadly subjective. Some users or modellers might be sensitive to what they might view as a 'distortion' of 'their' semantics.

Also schema integration involves articulating knowledge from various sources, and as such the whole process involves a great deal of irregularity as pockets of knowledge influence each other. This aspect is further complicated by the subjectivity of the knowledge. These include any descriptions/annotations, properties (the usual attributes) and any constraints defined upon the concepts. In addition, additional knowledge about the data model constructs is also used, such as the underlying meaning behind constructs such as attributes, entities and relationships. As stated in chapter 2, design can be considered as a process that produces a model of an object to meet certain requirements, such as compliance with a data model. This latter view of the process is syntactic, while the former is semantic. The fusion of this knowledge during schema integration manifests statements of equivalence/conflict between components of the input schemas. Indeed, most of this knowledge is embedded within the subschemas and is exposed during semantic/syntactic analysis of the subschemas or comes to light through deliberation.

Though schema integration is achieved through some technical paradigms, the underlying information on which it is based is socially-rooted and is very human-intensive. Some of the information resides in the heads of the modellers that constructed the input schemas because of the inherent lack of the EERM (and indeed data models in general) to represent all the semantics of concepts. Deliberation seems a prescription to offset these shortcomings. The next chapter discusses the social dimensions to complement the technical aspects of schema integration.

Chapter 6

Collaborative Schema Integration.

Summary.

This chapter endeavours to address the nature and requirements of CSCW (Computer Supported Cooperative Work) in schema integration. CSCW can be defined as the use of computers to support a number of workers working together on the same or related work to achieve a certain goal. Section 1 is an introductory section which defines the concept of computer-supported collaboration. Section 2 highlights the need for collaborative schema integration. It states the stages of schema integration that are amenable to collaboration and the supporting techniques that may be used. Section 3 addresses collaboration during preintegration i.e. concept selection and semantic analysis, while section 4 discusses collaboration during merging. Section 5 introduces deliberation as desirable for the process and proposes that individual mappings and their derivation need to be pooled together in order to establish semantic consensus among the stakeholders. Section 6 discusses how we envisage the deliberation mechanisms for collaborative schema integration. Section 7 concludes the chapter by stating the need for marrying CAISE and CSCW schema integration environments to store and manage integration models and their derivation.

1. Introduction: An overview of CSCW.

As the acronym implies, CSCW has the objective of supporting, via computers, human work arrangements that are collaborative in nature. The connotations of 'computer support' (CS) and 'collaborative work' (CW) are complementary in many respects [Schmidt and Bannon 1992].

CS focuses on the actual needs and requirements of people engaged in collaborative work [Schmidt and Bannon 1992]. In such situations, the computer and the person using it are viewed as a complementary partnership. Humans solve problems cognitively using the wealth of intelligence (natural or acquired) they are endowed with, yet are poor in managing the large volumes of data that is often associated with complex problems. Computer systems are however more suited to the latter by utilising storage mechanisms, checking and even presenting information in ways that aid understanding, thus amplifying human problem-solving capabilities [Fischer et al 1992]. It is therefore reasonable to consider computers as providing some support requirements for individual work. The notion of CW is concerned with a number of people working together towards a common goal. It deals with the

characteristics of collaboration such as communication, coordination, differences in work settings, solution principles, perspectives, responsibilities as well as interdependency with respect to timeliness and quality [Schmidt and Bannon 1992]. Figure 6.1 shows how bringing together these aspects manifest a CSCW environment.

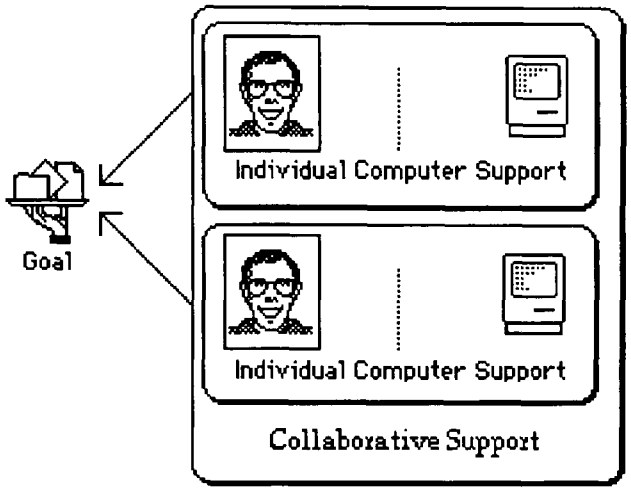

Figure 6.1: Goal-Centred 'Computer Support' and 'Collaborative Work'

Recently, there has been much interest in computer-supported collaboration (cooperation) for design work [Begeman and Conklin 1988, Fischer et al 1992, McCall 1988, Gronbaek et al 1993]. Such use of computers falls in the realm of CSCW (Computer Supported Collaborative Work) [Rodden 1991, Greif 1988] and underlines the fact that design in naturally collaborative. Argumentation systems are types of CSCW systems aimed at supporting and representing the negotiation and argumentation that is prevalent in group work [Rodden 1991, Fischer et al 1991(a), Fischer et al 1991(b)]. In addition argumentation systems allow us to capture and record the rationale for design decisions for documentation purposes. This rationale often includes the pros and cons of competing solutions to a design problem, all of which need to be clearly laid out for discussion. One fundamental problem with human deliberation is goal-setting. [Olson and Olson 1991] assessed time usage during meetings and found that "... *the first meeting involved a great deal of understanding the goal (e.g. understanding what was to go into the conceptual framework*". For schema integration, the goal(s) of each integration step need to be understood before the integration is actually implemented. In this view, CSCW may be considered to act as a common reference and focal point: putting constraints on information entered for

deliberation and then focusing deliberations. It is not surprising, therefore, that these systems have found their way into some design domains such as planning [Rittel and Weber 1973] as design by its very nature involves negotiation and deliberation [Rittel 1984]. However, it seems CSCW has so far not been investigated for schema integration despite its (schema integration's) human-intensive nature. Schema modelling and schema integration arise as a result of dialogues between users and data analysts and other data analysts. Such dialogue is necessarily collaborative in nature. Schema modelling does not involve much argumentation since data analysts rarely need 'argue' with the users who are supposed to be the experts in the information area. Instead it is much about negotiation than argumentation. However, there is much scope for deliberation in schema integration as integrators attempt to resolve any interschema ambiguities. Such resolution depends on understanding, which besides being subjective, has problems in *'how do you know when you have understood an object of study? If you looked a little harder or tried another angle, you might discover some hidden quality - a quality that reveals the object's true nature. Then again, maybe not'* [Rettig 1993].

[Rodden 1991] and [Rodden et al 1992] give a comprehensive discussion of collaborative systems and the supporting computer technology respectively. Distribution and sharing are major characteristics of a collaborative system. Distribution is a direct consequence of the need to support multi-user working environments. However, distribution brings along 'where' and 'when' issues to the interaction between collaborators. Figure 6.2 shows how these factors determine the possible forms of interaction. Generally, the interactions can be classified as either synchronous (all participants present at the same time), asynchronous (no simultaneous presence of participants) or mixed (both synchronous and asynchronous) [Rodden et al 1992]. Sharing is fundamental to cooperative work. A participant's work must at least be seen by other participants otherwise the whole idea behind collaboration is defeated. Sharing also brings along issues of ownership, access control, roles of participants and transaction mechanisms.

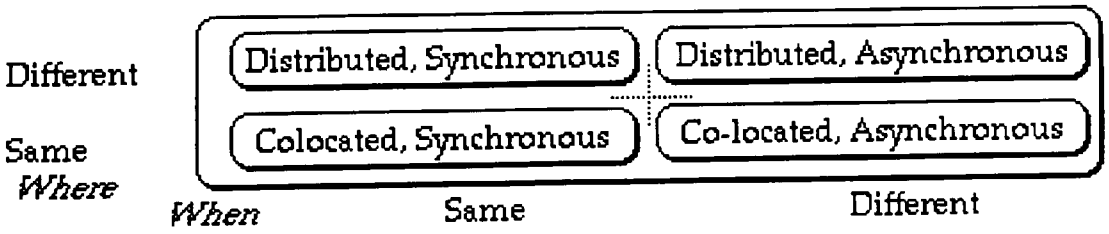

Figure 6.2: Time/Space Integration Matrix (adapted from [Rodden 1992]).

Asynchronicity allows working in parallel and to an extent avoids the dominance by one individual that is so common in face-to-face deliberation. A computerised asynchronous collaborative environment ensures that participants who would otherwise be dominated are given an equal opportunity to air their views, and entails that lack of participation due to protectionist tendencies towards one's subschema is curtailed. Such problems may be overcome by a system whereby once a user is presented with a request, a promise to service it and an indication of when the appropriate feedback might be due is given. Coordinator [Winograd 1987] and Information Lens [Malone and Lai 1987] are typical examples of such commitment systems. It also gives systems managers an opportunity to assess and realise the real potential of the individual participants to the organisation. However, knowledge that such information may be used by managers to assess one's performance might alter the participant's behaviour and even encourage dominance as a way to gain recognition.

2. CSCW and Hypermedia.

Hypermedia is widely recognised as an environment capable of supporting collaborative work [Conklin 1987, Fischer et al 1992, Streitz et al 1991, Irish and Trigg 1989].

The work of [Streitz et al 1991] was authored in a collaborative spirit: five of the leading gurus in the field were presented with questions by the moderator (Streitz) on a number of topical issues on this subject. There was no doubt among the panelists on the utility of hypermedia to CSCW in establishing a shared 'group memory'. The strength of hypermedia in this respect mainly lies in its ability to represent modular chunks of information and its ability to form an associative store (the group memory) populated with ideas, explanations, annotations etc. However, Ishii (panellist) advocates protocols of coordination in hypermedia-based groupwork environments. He argues for the need to control the flexibility associated with hypermedia and to capture the dynamic flow of discussion in groupwork settings. His comments on 'extending the link concept' seem to underline the importance of typing in hypermedia. Malone (panellist) echoes this idea when he advocates that formally representing certain information facilitates its processing (by computers or humans). Olson (panellist) adds that the potential advantages of hypermedia may be overshadowed by the overhead associated with its use.

The idea of applying hypermedia to collaborative work first appeared in Engelbart's Augment System, then known as NLS [Engelbart 1975]. It used a

component called the Journal to store designs and notes and from which they could be accessed by a number of authors. Nelson's Xanadu project followed along the same lines by offering linking to and annotating other people's work [Nelson 1975].

[Irish and Trigg 1989] give a comprehensive account of the issues that arise from observing use of a specific hypermedia development system, NoteCards, in collaborative work. Their work is interesting to this work because HyperCard is in many respects similar to NoteCards (some attribute it to 'Xerox envy'). They reiterate the suitability of hypermedia for supporting collaborative work. Hypermedia's use varied from commenting, annotating, discussing, sharing, coauthoring (by 'draft-taking'/'turn-taking') and storage of collaborator attributes (such as name, preferred font, text editor) to support collaborator-specific behaviour. Perhaps the closest to the intentions of this work was the group of collaborators working on instructional design for copiers: 'They *have discovered many common subsystems among these machines, and have pulled material relevant to these sub-systems into a generic subsystem notefile. They use cross-file links to point from the original notefiles to the generic notefile'*. This form of integration is however different from that for subschemas with respect to conflicts, system activity and knowledge requirements.

It is therefore not surprising to find that hypermedia has been applied to issue-based design, rationale capturing and structuring [Conklin 1987, Burns and Whitten 1990]. [Conklin and Begeman 1988] discuss the gIBIS tool that supports group communication through a graphic interface. [Burns and Whitten 1990] discuss a hypertext system, UIDD (User Interface Design Database), to capture and store the decisions and rationale of design decisions for user interfaces.

[Wood and Wood-Harper 1993] advocate decision-support technology that includes:

a. *' a focus on the formulation of the "problem" rather than merely providing an "objective" description of the problem'.*
b. *'... allow the decision-maker to explore the problem context in terms of constraints, likely impact, leverage, etc.'.*
c. *'... provide for "conversations about possibilities" ...'*
d. *'... evolutionary and emergent in nature'.*

A hypermedia environment for collaborative schema integration is a natural candidate for such decision-support: for (a) above, the 'problem' can be formally formulated in terms of assertions (statements of equivalence or conflict), for (b), (c) and (d) a hypermedia environment's node and linking mechanisms provide flexibility for such exploration, for the conversations about integration decisions and for the evolutionary nature of the conceptual schema. We discuss next the overlap between CSCW and schema integration.

3. The Need for Collaborative Schema Integration.

In a paper on software inspection [Knight and Myers 1993] quote Petroski as stating that:

'Engineers today, like Galileo three and a half centuries ago, are not superhuman. They make mistakes in their assumptions, in their calculations, in their conclusions. That they make mistakes is forgivable, that they catch them is imperative. That is the essence of modern engineering not only to be able to check one's own work, but also to have one's work checked and to be able to check the work of others' [Petroski 1985].

The truth of this statement for database design in general, and schema integration in particular, is unquestionable. They state, however, that such work is still limited by its dependence on human effort, and suggest supplementing the process with computerisation. Factors such as solution principles, restructuring for modularity and communication (during merging) are important, but by their very human-intensive nature are difficult to automate. Indeed *'every formulation of the problem is already made in view of some particular solution principle. If the idea of the solution is elaborated or even changed during the design process, new aspects become relevant and new kinds of information will lead to different questions about what is the case in the particular situation and about what is desired or acceptable. Since nobody can anticipate all conceivable design possibilities before design starts, nobody can list all potentially relevant data in a complete, well-defining problem formulation'* [Rittel 1971].

The preceding chapter introduced the idea that schema integration is basically a negotiation process due to following factors:

a. The inability of data models in general to capture all the relevant semantics necessary to establish concept equivalence and conflict. In this respect the

EERM has been further criticised for lack of semantic relativism [Saltor et al 1991], the ability of a schema to represent all the different conceptualisations of the same real-world object. The additional semantics are distributed among the data analysts who modelled the subschemas. Additional semantics may be drawn from beyond the scope of the subschemas, such as abstract knowledge about actual data instances and experience which often resides in the head of the individual subschema modellers.

b. The EERM has also been criticised for having more than one basic structure [Saltor et al 1991] (attributes, entities and relationships) that make it difficult to conceptualise about data from various constructs with varying syntactic weight. These multiple structures and their ability to model the same semantics in different ways complicate the formation of mental mapping models by a single individual.

c. Often, no one data integrator can simultaneously make a correct assessment of data semantics from the multiple perspectives that gave rise to the ambiguities. Also, multi-perspective knowledge requirements are likely to overwhelm a single individual, especially as schemas evolve.

d. a decision to integrate two concepts is only made after one has 'understood' the concept represented by some modelling construct. This is often the case when the subschema under observation was modelled by a different person as is often the case. Such comprehension is also subjective due to the inherent ambiguity in the meaning of linguistic symbols appearing in the subschemas e.g. how synonyms and homonyms arise from naming concepts.

e. Some integrations depend on others, implying some degree of interdependency between integration components of subschemas.

f. the gleaning of semantics is a learning process which may be aided by simple question-answer-clarification cycles. This informality is an important and everyday aspect of groupwork and often leads to persuasion, compromise and agreement.

g. One fundamental assumption about database design is that *'an outside expert or modeller can provide a formal description of the problem (domain) under consideration stems directly from the belief that such problems exist as "objective realities" ...'* [Hirschheim and Klein 1989]. However, this assumption has been questioned from the sociological point of view by many researchers who point to the relevance of a more subjective or interpretative account of organisational reality [Beynon-Davies 1992(b)]. In this approach reality is a socially constructed phenomenon. An

apt illustration of socially-constructed reality is given by [Brown and Duguid 1994]: *'The noise of a machine, for instance, is usually peripheral for most users, but it can be central for a mechanic.... When a machine malfunctions, its sound may move from the periphery of its user's attention to the centre'.* Thus the process of representation may even contribute to the construction of reality. If reality is objective then the representation can be seen as a filtering process. If reality is socially constructed then representation must be seen as negotiation. It is therefore reasonable to conceive conceptual modelling as utilising both forms of representing reality (subjective and objective) as data modellers switch from one to the other according to the problem at hand. However, integrating 'objectively' and 'subjectively' modelled subchemas that are semantically related draws in further human-dependency.

These factors may lead to integrations that are contradictory, incomplete or inaccurate. A contradiction may manifest itself when an instance of the integrated concept fails to represent a real-world concept corresponding to either of the integrated concepts. Their emergence in schema integration can be attributed to the fact that statements that are true in the abstract world are liable to be contradictory in detail. Incompleteness manifests itself in the form of absence of information e.g. mutual exclusion between values associated with a specific instance of an entity. Inaccuracy manifests itself when the integrity of data is suspect. Some of these anomalies may be automatically detected, but in most cases it is the user (and to a lesser extent the modeller of the concept) who is the best arbiter of a contradiction, completeness or accuracy.

[Chen et al 1989] note the lack of CAISE tools that enhance and focus interaction between stakeholders of a system under development. They suggest that because most tools do not support human interaction some users are reluctant to use CAISE tools. During schema integration stakeholders interact to communicate and elicit knowledge, create alternatives and resolve ambiguities. [Leite and Freeman 1991] aptly put it:

'The principle that more sources of information provide a better understanding of a subject has been used for centuries, in court investigation, for example. Different witnesses may have conflicting or complementary recollections. By using this principle ... the chances of mastering correctness and completeness problems will be greater'.

Thus an environment for collaborative schema integration may provide some solutions to some of the cited problems. We propose below how such collaboration may be realised during schema integration.

It may be envisaged that the integrators would work together in identifying ambiguities and submit an integration proposal for deliberation with other integrators (and users) so that an ambiguity can be established as such and the resultant unified representation of the data can be established based on semantic consensus [Baskerville 1993]. To reduce the significant duplication of effort by a constant reinvention of the wheel, it is desirable that all assertions are unique insofar as all the participants are concerned. The ensuing deliberation would be basically argumentative, and aims at critiquing the integration proposal from the many perspectives represented by the modelling constructs in question. Often such argumentation triggers deeper analysis of the concepts to be integrated with the result that some useful latent semantics may be discovered. It is obviously advantageous to discover such semantics early otherwise more work will be required to incorporate them later in the life of a system. An advantage of such an approach is that most of the information captured in the integration proposals pertain to establishing the rationale of an integration decision. Such information is invaluable during the life of an organisation's database development and administration.

It is reasonable to consider that once an integration proposal has been proposed it goes through a number of states until it is resolved. Figure 6.3 shows the transitions through which such an integration progresses. Initially the proposal is in a 'pending' state during which it is constructed. It then goes into a 'submitted' state, in which case it is ready to be deliberated upon. Then due to a participant's desire to critique it, it goes into a 'reviewing' state, during which the participant in question attempts to understand the proposal and reconcile it with any additional knowledge at his/her disposal. From the 'reviewing' state, a participant may decide to:

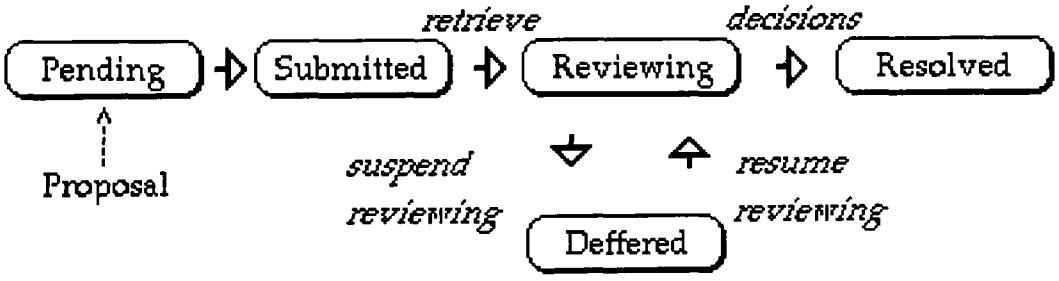

Figure 6.3: States and Transitions of an Integration Assertion.

a. submit his/her critique ('submitted' state), often channelling them to the proposer.
b. defer reviewing pending the availability of further knowledge ('deferred' state).
c. resume reviewing using the available knowledge (back to 'reviewing' state).

Since there are multiple participants, each proposal may undergo multiple 'reviewing' and 'resolved' states. For the transition to the 'resolved' state the participant has to provide the rationale which is stored in the integration dictionary. Note that there is no 'rejected' state as we want a proposal to be critiqued as much as possible. The final resolution of the proposed integration is then decided upon based on the 'decisions' about the proposal. Such a decision is taken by a person who assumes the role of 'facilitator'. Such a person is likely to be the database administrator (DBA) of the organisation.

4. Collaboration During Schema Integration.

It is virtually impossible to identify a stage of schema integration in which collaboration is not needed. However, we can state with certainty that some stages are more collaboration-intensive than others. For instance merely renaming an entity during merging does not require as much deliberation as the process of establishing homonymous names. Figure 6.4 gives an overview of the usage of collaborative integration, showing where schema integration and collaboration aspects overlap. Figure 6.5 shows the proposed collaborative schema integration strategy.

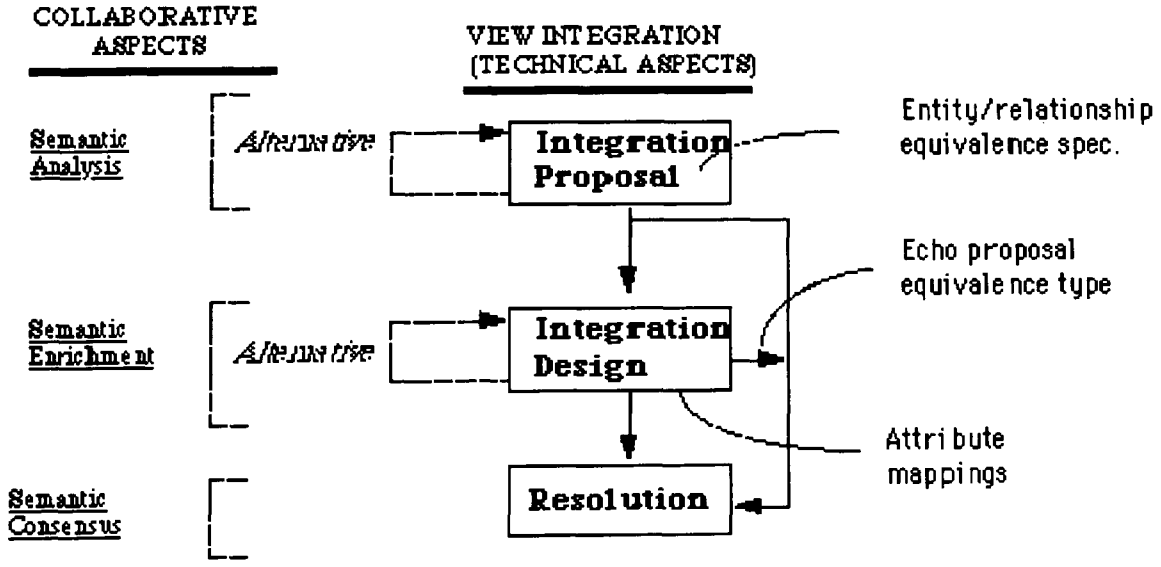

Figure 6.4: Schema Integration Tasks Requiring Collaboration.

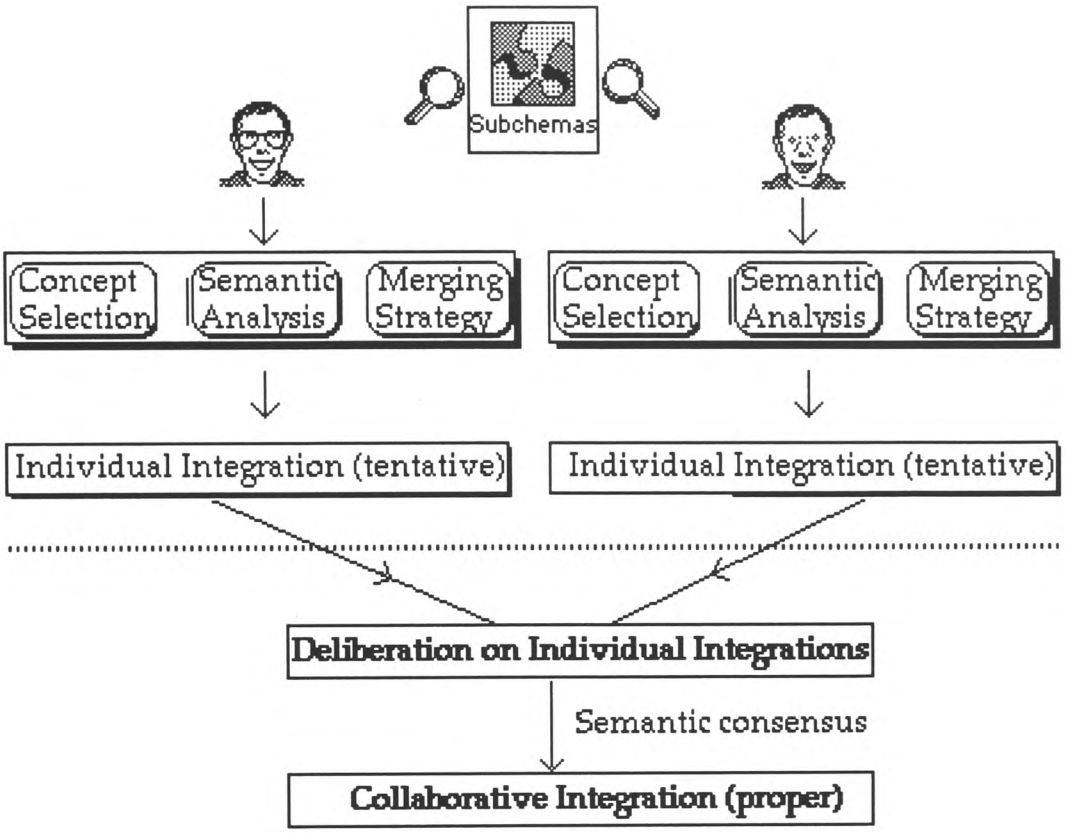

Figure 6.5: Collaborative Integration Strategy.

We illustrate some traps associated with schema integration and show how collaboration is used to reduce their effect, provide alternative integrations and fine-tune integrations. Figure 6.6 shows an integration scenario for two schemas in a university situation: one is for a department and the other is for the enrolment office. Data analyst A produces the schema SC1, while B produces an alternative schema SC2. Integrators A' and A'' examine the schema SC1 and find out that there are two paths from the entity 'Student' to the entity 'Department'.

They ask, for instance, if the relationships (Department enrolls Student) is equivalent to the relationships (Student takes Course) and (Course offeredBy Department). This question might already have been asked by B in getting SC2. Convinced they are not, A' resolves this conflict by renaming one 'registrationDepartment' and the other 'teachingDepartment' thereby getting the same schema SC2 as B. Alternatively, A'' is intent on retaining the entity 'Department' so he/she decides to specialise it with the entities 'registrationDepartment' and 'teachingDepartment'. Meanwhile, observing the overlap between 'registrationDepartment' and 'teachingDepartment' in schema SC2, integrator B'' produces the same abstraction.

This example illustrates some of the options open to integrators during schema integration: integrator A's option is inappropriate (a department teaching a student does not necessarily enrol him). Further it illustrates how many sources of knowledge are likely to produce better results. A' and B produced schemas that represent the situation more closely, while A'' and B' opted for a more abstract representation. For the sake of brevity, this illustration does not show the decisions that led to the given integrations and it is reiterated that capturing them is as important as producing a good integration.

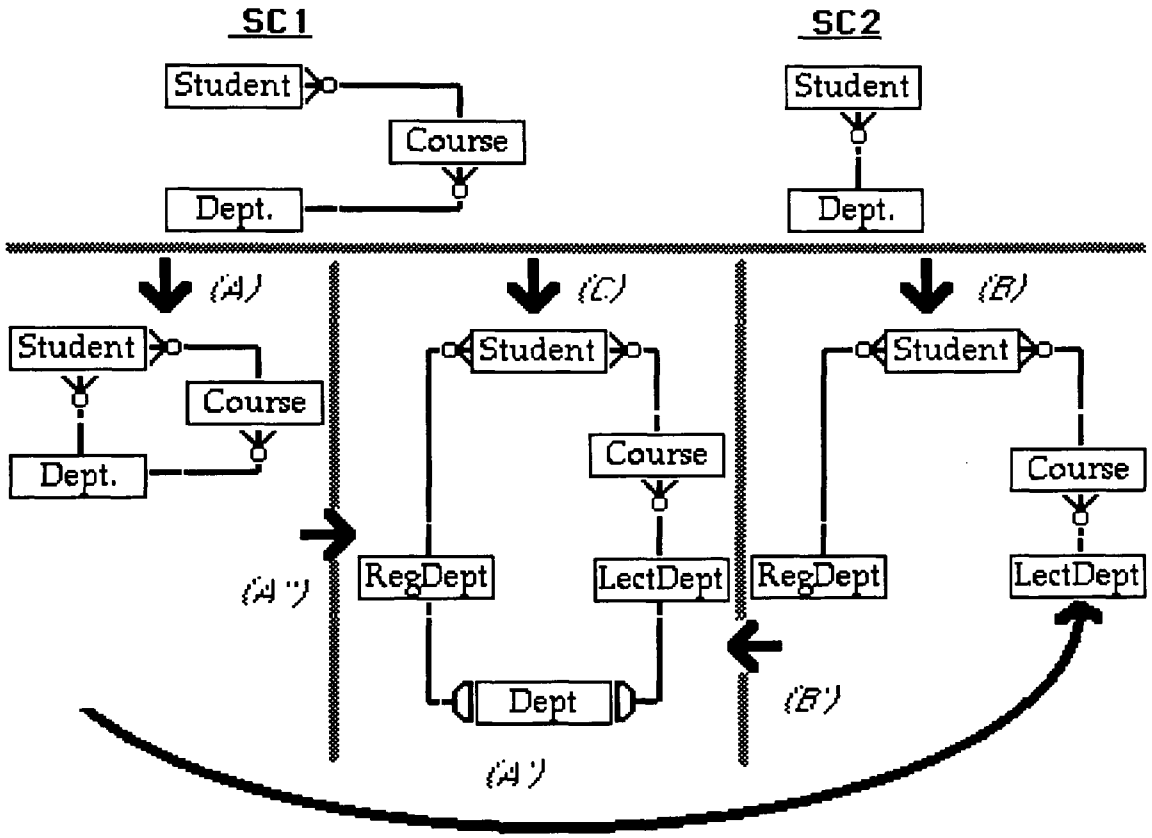

Figure 6.6: An Illustration of Multiple Schema Integration Solutions.

4.1. Collaboration During Preintegration.

Preintegration comprises the stages of concept selection, equivalence and conflict analysis and specification. Selecting which concepts to integrate may influence, be influenced by and determine a number of factors during the integration process. The process of reading (specifying) other's (one's) assertions and annotations for some integration is indeed a collaborative contribution to the work. The following subsections discuss collaboration in these stages of preintegration.

a. Concept selection.

The concept selection stage is ignored by most of the literature despite its vital importance in determining the rate of integration convergence and the quality of an integration. A case in point is abstraction as a measure of the quality of a conceptual schema (data analysts, like all systems professionals, work in abstract terms). Good abstractions are identified bottom-up because of the constructive nature of the bottom-up approach, unlike the top-down approach in which the boundaries of a concept are predefined. So an integration in the order entity-attribute, entity-entity, entity-relationship and relationship-relationship is desirable. Abstracting data is a vital design technique during schema integration. However such abstractions must not hamper data visibility and communication which are vital for understanding the semantics of concepts.

The few works that do discuss concept selection are based on ordering concepts with respect to some weight measure. [Batini and Lenzerini 1984], for instance, suggest picking concepts for integration in decreasing order of their relevance. For example, partially integrated concepts may be preferentially selected because they are important, or because they naturally follow an integration trail to completion. Some concepts may be preferentially selected on the basis of a relative evaluation of the importance of its many users e.g. selecting concepts in the 'enrolment' department in preference to the 'accommodation' department.

One problem with this approach is that it is difficult to associate weights with concepts, especially those concepts that are at the same level. Besides, even if it were possible, these weights are likely to vary after each integration thus requiring reassignment. This is however not very pragmatic unless weight assignment is done 'automatically'. Other factors that may influence concept selection are factors such as the anticipated abstraction level of the integrated concept, the anticipated rate of convergence and the volume of ambiguities which are human-intensive. Because integrations tend to result in the establishment of new entities or relationships, it might be reasonable to start integrating concepts with the greatest equivalences/conflicts in the hope of quickly reducing the volume of ambiguities in the subschemas. Integrating two abstraction hierarchies may best be done in a top-down fashion because of pragmatics- a bottom-up integration of subentities would involve importing all inherited attributes first.

If a participant has no queries about integrating the selected concepts then he/she goes into 'reviewing', otherwise the integration proposal is 'deferred'

after which the participant elicits further information e.g. by questioning. When the information on which the deferred proposal/issue is dependent has been provided, the deferrer may resume contributing towards the proposal or issue.

b. Equivalence and Conflict Analysis.

Schema integration is indeed a decision-making process. According to [Wood and Wood-Harper 1993], the process of decision-making can be considered to blend with the highly cited 'intelligence-design-choice' model [Simon 1977] and authorisation [Mintzberg 1976].

Intelligence is defined as the process of gathering information by exploring the decision domain. Foremost in this process is learning: the integrator must understand all the relevant information. We in fact expect these requirements to be met in our subschema hyperdocuments by way of structuring and presenting subschemas in comprehensible ways (modularity, flexible access, information hiding etc.). For instance the integrator goes to an entity node and gleans the semantics of the entity and selectively traverses a relationship path to an abstracting entity or a relationship in which the entity is a member. Secondly, the intelligence gathered is stored in an integration dictionary which has both immediate and historical value. For instance the intelligence gathered would document why the age of a person is derivable from the person's date of birth and the incompleteness (with respect to date of birth) that would result in the event of age being the final representation.

Designs are the proposals of possible integration solutions and may involve the creation of additional semantics. While design creates integration solutions, choice evaluates the alternative solutions. Intentionally, the alternative design may be stored as a self-contained proposal: yet another exploitation of hypermedia.

Authorisation ensures that of all the given alternatives and their associated documents, the facilitator chooses an optimal choice. The optimal choice must be convincing to the stakeholders, hence it must be communicated back to them together with the rationale for its choice. The rationale is documented in the form of an annotated decision trail. The stakeholder is thus empowered with the information relevant to convince him/her of the validity of the solution to an integration proposal.

c. Equivalence and Conflict Specification.

An equivalence or conflict specification may include its explanation. However, the specification itself or its explanation may be inaccurate or even wrong, calling upon other stakeholders to respond it. Some may decide to agree or disagree with it, while some may opt to improve or scale down the integration. Whichever way, the stakeholders contribute to the specification by formally stating their opinions about the equivalence. This can be considered an evaluation exercise for the specified equivalence and potentially involves the greatest deliberation.

5. Collaboration During Merging.

As discussed in Chapter 5, concept merging involves concept transformation. Having identified two concepts as equivalent or conflicting, thanks to the equivalence and conflict analysis stages, the integrators transform one of the member concepts of an integration by adding, deleting or moving attributes to the other, or by renaming the other in case of a naming conflict. There are a number of factors that affect transformation decisions, dependent on the individual intention of the integrator. These include:

a. Transformation for removing naming conflict: In case of synonyms, the transformation involves removing the duplicate concept, and for homonyms appropriately renaming one of the concepts. It is not expected though that this simplistic task requires deliberation, as long as everyone gets to understand that instead of calling it 'A' we now call it 'B'.
b. Transformation for attribute allocation: This task enriches or partitions concepts by adding, deleting or moving attributes between entities and relationships. Consider the case of a monogamous marriage: is property/child an attribute of the husband or wife? The many alternatives to these questions point to a possible resistance to such attribute allocation due to the inherent 'implied' dominance or of loss of control that may result. A democratic solution to this problem would be sharing i.e. converting the relationship into an entity to which both spouses are related.
c. Restructuring to present particular data aspects: The problems associated with this aspect are the same as those faced by a data analyst in deciding whether a concept should be represented as an attribute, an entity or a relationship, or even a hierarchy. Issues about semantic dominance are drawn in, but more importantly are issues pertaining to abstraction hierarchies. Abstraction hierarchies are powerful in representing particular

data structures, such as a student and a lecturer being both specialisations of the entity 'person'. Bearing in mind that attributes are the embodiments of data, this task is also responsible for changing the abstraction level of a concept. But the abstraction level has great significance to how data is modelled, and the significance of the data. A concept at the attribute abstraction level is considered to be of lesser 'semantic' significance than a concept at the entity abstraction level, the idea being that an entity aggregates different but related attributes. Consider again the marriage concept. A marriage can be modelled as a relationship between a man and a woman (depicting man and woman as concepts of equal syntactic weight!), or considering man/woman as an attribute of woman/man (certainly depicting some semantic dominance). This example clearly shows that the final representation is potentially a source of conflict which needs deliberation. Some abstractions though organisationally valid may not be relevant to the individual information areas. Consider the generalisation of the entities 'secretary', 'technician' and ' lecturer' by the entity 'employee' is significant because all these entities relate to the same organisational aspect: employment by a 'university'. However, generalising 'student and lecturer' with 'person' seems to be organisationally insignificant (i.e. person does not store anything of interest to the organisation), unless otherwise all persons are considered major entities in the organisation.

6. Deliberation Mechanisms.

Schema integration can be considered a decision-making process. A computerised decision-making process has to be supported by a computer-based model [Gordon and Fry 1989]. That subschemas may overlap is inevitable, but that redundancy is eliminated is imperative. The schema modellers therefore need to share their schema and integration ideas. Doing so ensures that semantic consensus about data satisfies all the stakeholders. However, getting semantic consensus may involve a great deal of deliberation. In addition to the deliberation mechanisms of the IBIS scheme (issue, position and arguments), deliberation may use techniques such as questioning, answering, annotation, commenting, explaining and data scenarios (examples and counter-examples) to complement the weaknesses of data models. Such additional information is very useful in expressing an incompleteness, an inaccuracy or a contradiction. A counter-example is often used as a way to prove incompletenesses and inaccuracies. Semantic consensus also involves the evaluation of alternatives. This can be achieved through a number of

techniques to which hypermedia has been applied in other fields, such as annotated trails (of decision influences), guided tours, voting systems and authorisation systems.

7. Conclusion.

In this chapter we have discussed computer-supported collaboration for schema integration. We started by briefly discussing CSCW and hypermedia support for it. We have addressed some schema integration problems and highlighted the need for collaboration to address them. Also given is an outline of the technology to support such collaboration: a hypermedia environment utilising the IBIS scheme.

The main theme emerging from the chapter is that the gleaning of semantics and the attainment of semantic consensus is collaborative in nature. Hypermedia can be used to flexibly provide the 'minimum' cognitive overhead during semantic analysis by providing concise information. It also supports the maintenance of partial schemas and their supporting documentation as well as supporting the collaboration itself. The next chapter discusses a prototype hypermedia-based schema integration tool developed to demonstrate these features.

Chapter 7

A Prototype System for Collaborative Schema Integration.

Summary.

The previous chapter discussed some salient features of a collaborative schema integration environment and highlighted hypermedia as an enabling technology for such work. In this chapter we discuss a working prototype of such an environment. The prototype is by no way complete but it is maintained that the work has provided sufficient insight into the efficacy of such environments. Section 1 gives an overview of the system, discussing its context. Section 2 discusses the architecture of the system and an adaptation of the IBIS (Issue-Based Information System) scheme [Rittel and Kunz 1970]. Its links to the HSMS (Hypermedia Schema Modelling System) (c.f. chapter 4) are discussed as is the management of integration models. Section 3 proposes a model for a hypermedia-based IBIS environment for collaborative schema integration. Section 4 discusses the prototype's support for the preintegration phase of schema integration. The deliberation and merging facilities of the prototype are the subjects of sections 5 and 6 respectively. Sections 7 and 8 discuss the prototype's analytical and browsing facilities respectively. Section 9 gives an illustrated example of the use of the prototype in integrating sample schemas. Section 10 compares and contrasts the prototype with other such systems. Finally, section 11 lists the conclusions arising from the work on the prototype.

1. Introduction.

Performing schema integration on real-life schemas without CAISE tool support can be very difficult, tedious and error-prone [Sheth et al 1988]. Many CAISE tools for schema integration have adopted an automated approach whereby a tool automatically performs the schema integration based on some built-in heuristics [Yao et al 1982, Bouzeghoub and Comyn-Wattiau 1990, Hayne and Ram 1990]. Usually the tool optionally allows the data integrator to confirm the decisions it suggests [Elmasri and Navathe 1984, Navathe et al 1986, Hayne and Ram 1990]. The expertise built into such tools often analyses the subschemas in search of ambiguities. While in sympathy with this approach, we argue that this approach has the following problems:

a. automation has shortcomings because of its lack of consideration, and support, for the negotiation that underlines schema modelling and schema integration. Firstly, we need to consider the negotiation aspects of schema

modelling as schemas are negotiated abstractions of real-world domains. Secondly, we need to consider negotiation for schema integration such that our representations are based on semantic consensus.

b. schemas, as abstraction mechanisms, often lack the detail necessary to ascertain equivalence. For instance, background assumptions and information which are naturally part of the negotiation process are missing. Thus it may be impossible to have enough knowledge to automatically determine equivalencies and/or conflicts. Such knowledge needs to be gleaned from the modellers of the subschemas or even the users. It may help clarify some assumptions taken during the modelling process.

c. relationships between real-world concepts do not merely depend on the relationships between the names of their representations in the subschemas. Consider, for example, the 'equals' equivalence assertion. A name-based topological view of this assertion states that for every attribute x in A, there exists exactly one attribute x in B, and vice versa. While this may be true in certain circumstances, this view is extremely dangerous as it totally ignores the perspectives to the data. Parallels can be drawn with the view concept in relational databases: a view is derived from a relation(s) and each view gives a different perspective to the same relation(s) but this may not (and need not to be) be reflected in the names of the views.

For these reasons and the arguments we provided for collaborative schema integration in Chapter 6, a human-centred approach was taken in developing SISIBIS (Schema Integration System with IBIS). In the human-centred approach knowledgeable integrators forward the equivalence/conflict assertion to the system.

The essential task of SISIBIS is to support the development of an equivalence/conflict model comprising equivalencies/conflicts among components of subschemas being integrated. It is an interactive tool which collects integration information from the integrators (sometimes referred to as participants in the text). In some respects it is similar to the tools discussed by [Sheth et al 1988] and [Yao et al 1982]. It allows participants to propositionally assert the equivalencies between concepts in the subschemas. The assertion is specified as a formal statement. The tool goes further by managing the issues arising from collaboration in an issue-base constructed using an adaptation of the IBIS framework. In so doing it captures the rationale behind an integration proposal or decision. SISIBIS uses HyperCard 2 as the underlying hypermedia system and utilises some of the facilities offered by HSMS (c.f. Chapter 4). It

allows both synchronous (over a network) and asynchronous collaboration (by turn-taking on the same machine). This tool can be considered to be one small instance representative of Yourdon's vision for CAISE in the 1990s: *'it is quite likely that the most significant development of the mid-to-late 1990s ... is an new generation of groupware CASE tools. As the term implies, these tools will be more concerned with the activities of groups of people, rather than focusing on the needs of individuals'* [Yourdon 1993].

2. Architecture of SISIBIS.

Figure 7.1 shows the basic architecture of SISIBIS. It employs an integration dictionary, a preintegration facility and a merging facility. The integration dictionary consists of the subschemas, the collaborative integration facility and the evolving schema. The collaborative integration facility is home to the proposals and the issue-base resulting from deliberation. It manages the assertions and the explanations, comments, data scenarios, questions and answers pertaining to the assertion. The preintegration facility offers selection and browsing of input schemas and the concepts to integrate, while the merging facility offers resolution and implementation of the integration. The merging facility also offers a utility to link the rationale to the resolution node of the integration proposal/issue. Further, the merging facility provides the ability to append to the integration schema all the unintegrated concepts of the subschemas, thus establishing a partial integrated schema.

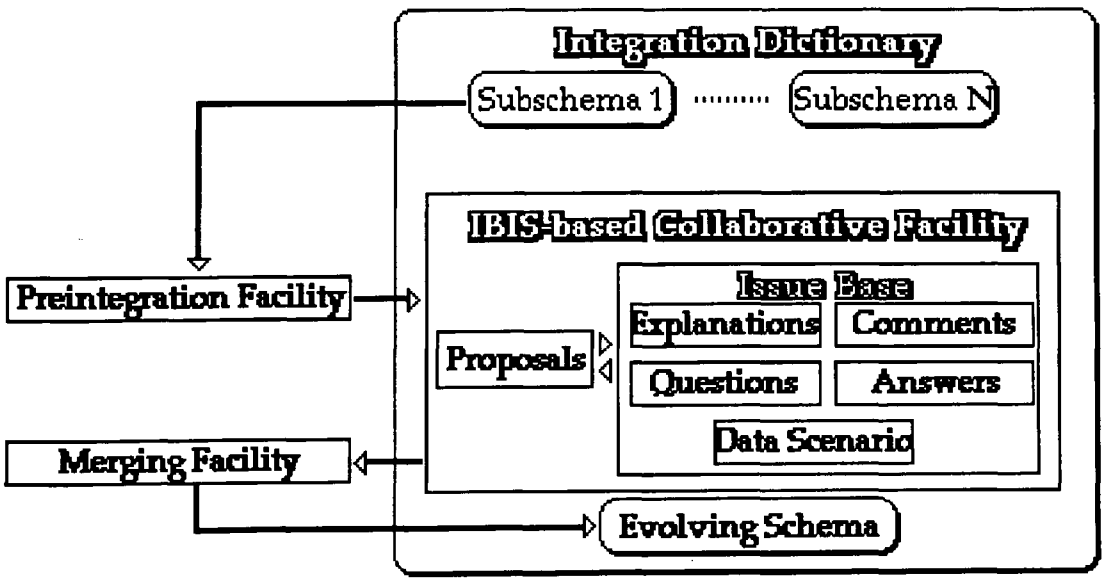

Figure 7.1: Architecture of SISIBIS.

It would be desirable to have only one instance of the integration dictionary, but due to the limitations of accessing and modifying shared HyperCard stacks on a network the local-copy option was taken. Though stacks can be shared over a network, some participants may be denied access once someone has opened it, and for those allowed access to it, modification rights belong to the first person who opened it [Goodman 1990]. Considering that idea-generation is a lengthy process, the copy option may suffice in practice. Local copies are made on the instruction of the participant, but copies of the subschemas are made automatically after each update of the shared integration dictionary.

The main function of SISIBIS is to create and manage the integration dictionary. To create the integration dictionary, an integrator selects the subschemas to integrate according to a binary strategy e.g. University and Polytechnic. Participants may then populate the dictionary with assertions, to which others respond, in a deliberative manner, by attaching their responses in the form of annotations. These then constitute the issue-base for the integration. Based on the elicited intelligence, an optimal choice is made and the merging facility implements the integration. For instance, after deliberating over the equivalence of University.Course and Polytechnic.Course, the choice may be to 'Union' them.

3. A Hypermedia-IBIS Environment for Collaborative Schema Integration.

The major questions in collaborative schema integration are:

a. How to get integration ideas out effectively?. An environment that augments the learning of schema semantics is desirable. Equivalence/conflict identification requires knowledge about the subschemas and the information areas they supposedly represent. Acquisition of such knowledge may be aided by:

   i. flexibility in navigation within and across subschemas.
   ii. eliciting undocumented information from the modeller of the subschemas in clear and unambiguous ways.
   iii. documenting the knowledge (c.f. (b) below).

b. How to get the integration ideas into some form of database?. The ideas have to be pooled and stored, and a database of such ideas is likely to be semi-structured.

c. How to logically and intelligently organise the integration ideas?. With possibly a number of options for representing and interpreting concepts, the ideas need to be organised so that they do not overwhelm the reader.

Foremost, we need a model for idea gathering, representation and management. The IBIS (Issue-Base Information System) [Rittel and Kunz 1970] is a framework suitable for idea generation and gathering. A hypermedia environment supporting the IBIS scheme is suitable for representing and managing the gathered ideas (c.f. Chapter 6, Section 6). The IBIS method *'is based on the principle that the design process for complex problems is fundamentally a conversation among stakeholders ...'* [Begeman and Conklin 1988]. SISIBIS represents each integration proposal as a node. Any annotation (explanations etc.) associated with the proposal is also represented as a node. It is up to the participant to edit the annotations as self-contained idea units. The linking mechanisms of hypermedia are then used to link the relevant ideas in defining the rationale for an integration.

The IBIS scheme's aspects and our schema integration equivalents for them are:

a. **Issue:** An issue represents something about which deliberation is required to solve a specific problem. For schema integration the problem is to establish whether two modelling constructs represent the same real-world objects. Such an issue manifests itself in the form of a proposal for entity/relationship integration (higher level concepts) or of attribute integration (lower level concepts). Thus proposals for attribute integration may be considered as subissues.
b. **Position:** A position is a proposed solution to an issue. For schema integration we consider a position to consist of two aspects: why the concepts are integrable (i.e. the equivalence type) and how they are to be integrated (i.e. the merging method) (c.f. Chapter 6, Section 3.1.2(b)). A participant takes a position based on his/her knowledge of the real-world concepts.
c. **Argument:** An argument is a statement that provides the pros or cons of a stated position. For schema integration arguments provide the participants with an opportunity to specify their knowledge of the real-world concepts most of which might not be apparent in the subschemas. An argument manifests itself in the form of an explanation, a comment, a data scenario, a question-answer pair or an explained statement of agreement or disagreement to a position. An explanation is defined as a statement

describing the validity or invalidity of a position (proposal). A comment expresses a participant's qualifying or characterising opinion towards an integration proposal or issue. For instance, a participant conscious of business rules may add that while an integration is valid, its validity may be suspect when an additional information area comes into existence in the organisation. A question is a statement that elicits additional information, while an answer provides it (the information). Data scenarios are examples of real-world values showing the equivalent or contradicting data.

d. **Resolution:** A resolution is a choice selected from amongst the stated positions. The choice is influenced by the arguments provided for the position. It is hoped that, based on equivalence resolutions, SISIBIS can assist in consistency checking and validation. For instance, an entity/relationship integration proposal must be echoed by integration proposals of key attributes.

In order to adapt the IBIS scheme for schema integration, additional aspects (explanations, comments, questions, answers and data scenarios) were added. Figure 7.2 shows an E-R diagram representing the underlying model for the adapted IBIS-scheme's aspects. The italicised entities represent aspects added to enhance collaborative schema integration work. Each entity in the diagram is submitted by a stakeholder (not shown in the diagram) whose semantic analysis is perspective-dependent. Most of the entities on the many side of the relationships are due to the fact that there may be a number of stakeholders. Each stakeholder, however, may have one such entity instance for every instance of the other. This ensures that no participant argues or contradicts himself and reduces ambiguity in explanations, comments, questions and answers.

SISIBIS tags each of these constructs with the participant's name and also time- and date-stamps each proposal and its associated documentation. The name tag is used to identify the participant so that comments and questions are forwarded to him/her and the submitters of explanations, answers and data scenarios are known.

Figure 7.3 summarises the logical steps to be followed to achieve schema integration using SISIBIS:

a. the DBA (Database Administrator) selects the subschemas to integrate.
b. participants then select concepts to integrate from the subschemas, specifying assertions and then deliberating over them.

c. the DBA resolves the proposals by one of the alternative integration strategies proposed.
d. the DBA defines the rationale for the selected choice.
e. the DBA commands SISIBIS to implement the merger.
f. after all allowable integration proposals have been processed through (b) to (e) above, the DBA commands SISIBIS to append any unintegrated concepts to the evolving schema.

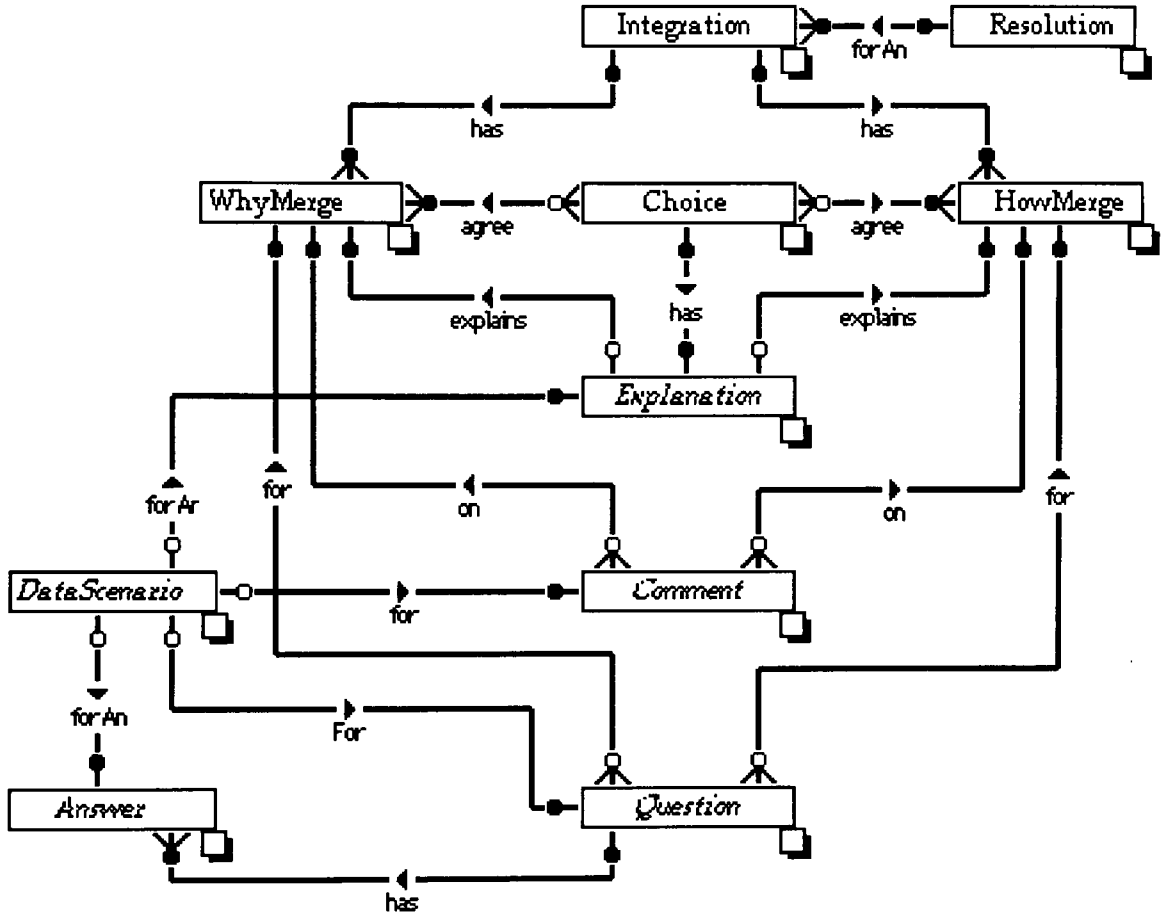

Figure 7.2: ER Diagram of SISIBIS Aspects.

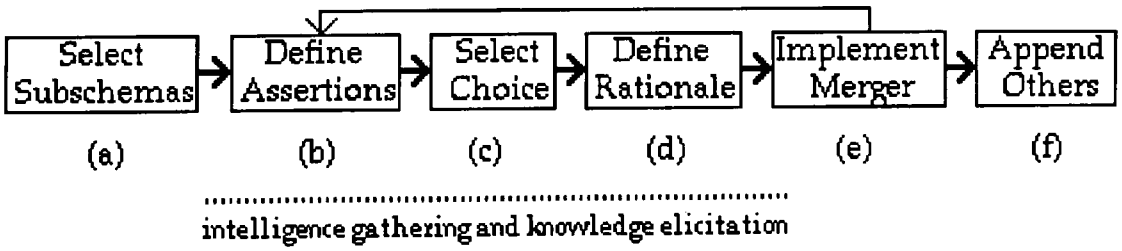

Figure 7.3: Logical Steps for Schema Integration Using SISIBIS.

Iterating stages (a) to (f) above manifests a new level of the binary-integration strategy at each iteration step. Iteration is deemed necessary under the following two conditions:

a. some integration proposals were rejected because an entity or a relationship may participate in only one integration proposal in any given issue-base.
b. the DBA intends to preferentially integrate a number of subschemas.

We discuss these steps and their associated activities in the following sections.

4. The Preintegration Facility.

The preintegration facility offers concept selection and browsing for equivalence and/or conflict analysis. We distinguish, in order of specification, three levels of integration:

a. that between schemas is termed a *project.*
b. that between an entity and /or a relationship is termed a *proposal.*
c. that between attributes is termed an *issue.*

Thus an issue is an integral part of a proposal and a proposal is an integral part of a project.

In defining a project, the DBA selects two subschemas to integrate. Such a specification may be based on knowledge of the overlaps between the information areas represented by the subschemas. Alternatively it may be based on an observation made during browsing the subschemas. To achieve this the DBA invokes the 'Create Integration Schema' menu. The facilitator is presented with a file-selection dialogue box through which to select the HSMS dictionaries of the subschemas. It then prompts the facilitator for the name of the integration schema and checks for any duplications. SISIBIS then creates the integration dictionary from a template and makes local copies of the subschemas to avoid problems in distributed access.

The integration dictionary is appropriately given the extension '.Σ' to signify summation. Figure 7.4 shows a sample integration dictionary's components. The subschemas 'Polytechnic' and 'University' are integrated into the evolving schema 'College'. SISIBIS then invokes the 'Open Integration Schema' menu with the created schema's name, after which the participant is taken to the integration stack. The participant may choose to create a new proposal or issue node via the appropriate buttons. Issue nodes are however

created only in the context of a proposal i.e. the proposal must exist first. Conversely, proposals and issues are deleted via a 'Delete' button, and deleting a proposal results in the deletion of all its issues while deleting an issue results in the deletion of its associated documentation e.g. explanations. The dictionary can then be populated with assertions. Hopefully the facilitator begins the deliberation by giving a proposal that initiated the subschema's integration.

The participant then selects the concepts to integrate by populating the formatted node shown in Figure 7.5. SISIBIS automatically enters the participant's name, the date- and time-stamps for the node and the counts for the assertion. The status bar indicates that the proposal/issue is 'Pending'. The form is ready to be populated with assertion information.

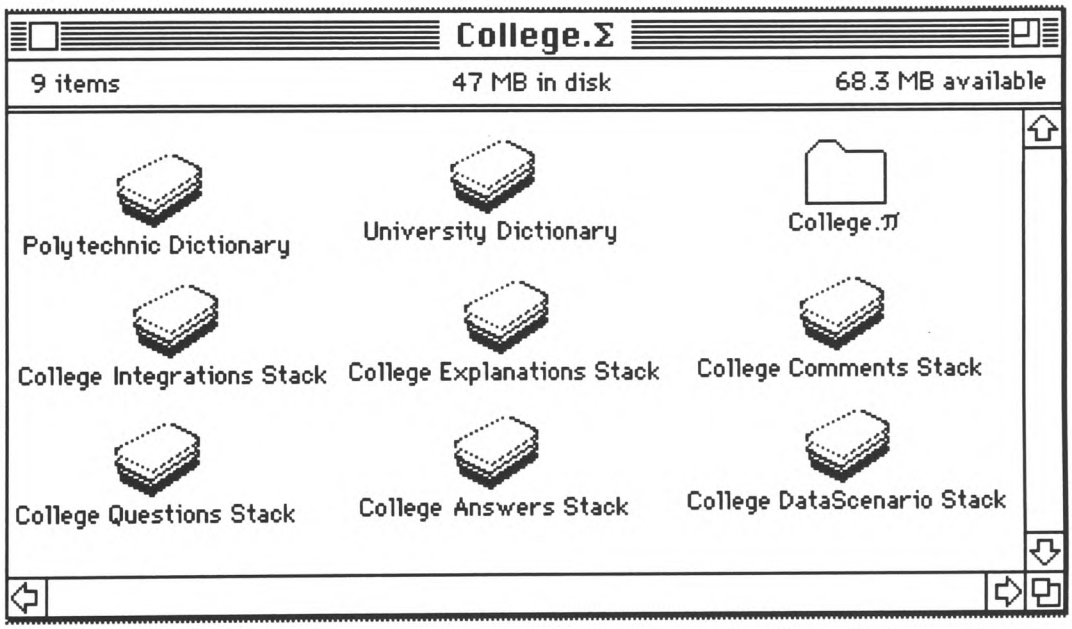

Figure 7.4: A Sample of SISIBIS Components.

To formally specify an assertion, the screen is divide into two areas: one for each of the subschema's information with templates for the subschema name, the modelling construct type (attribute, entity, relationship) and the actual concept(s) selected from the subschema. One of the two concepts represents the domain (the independent variable of a mapping) and the other represents the range (the dependent variable of a mapping). The specification relies heavily on pull-down menus. Pressing the mouse over the object type icon displays a pull-down menu offering two selections: 'Entity' and 'Relationship'. For instance, the figure shows 'Entity' as the construct type selected for both

subschemas. Pressing the mouse over the object selection icons also displays a pull-down menu listing the entries of the appropriate type (attributes, entities, relationships) obtained from the corresponding subschema. Concepts can then be selected or deselected from the list. The figure shows the entities 'underGraduate' and 'postGraduate' selected for the integration proposal. Since in any mapping, we map values from a domain to a range, we insist that the range is simple i.e. there is only one concept in the range. The domain may however be composite except for attributes. Thus we can define a mapping from the domain concepts (FirstName, MiddleName, LastName) to the range concept (FullName). Such a restriction makes the creation of mental models more manageable as only one (the range) has to be constructed based on knowledge of the domain. Further we allow such n-ary integration for attributes since we consider attributes to be atomic, hence the knowledge requirements to comprehend them are minimal compared to entities and relationships.

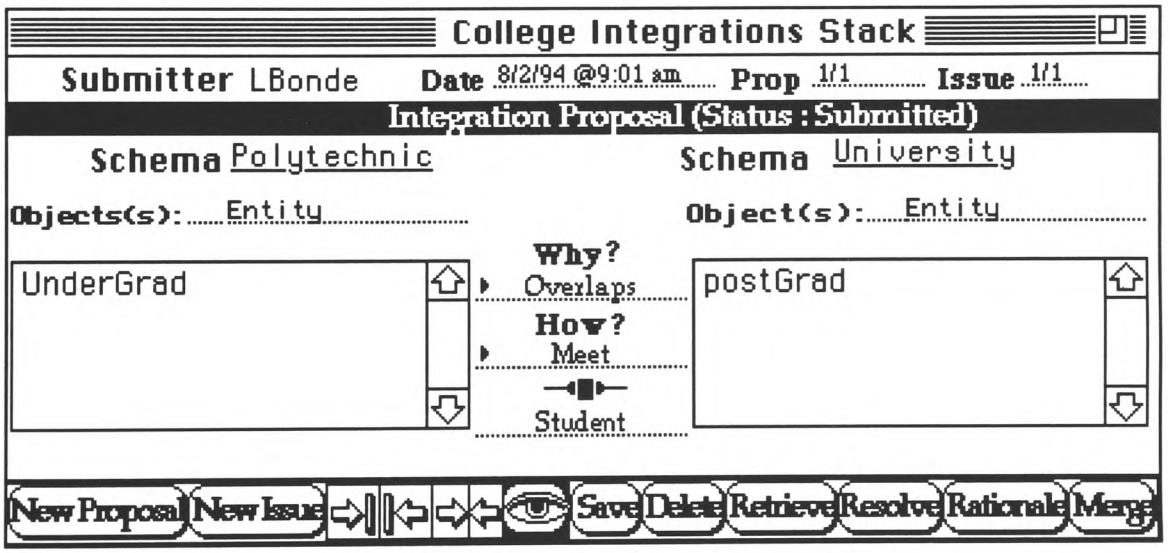

Figure 7.5: A Sample Assertion (proposal) Entry.

The participant then specifies the equivalence type (the 'Why') of the assertion. Invoking the 'Why' icon pulls-down a menu displaying the following equivalence types: Equal, Contains, Contained-In, Overlaps, Disjoint, Derived-from and Derives. The last two are in addition to those discussed in Chapter 5 and are meant to cater for derived attributes e.g. differences in scales (such as currencies) can be unified by means of a formula specifying the derivation. Figure 7.5 shows the equivalence type of 'Overlaps' selected.

To complete the assertion, the participant specifies the intended integration

method (the 'How'). This is achieved through invoking the 'how' icon, resulting in the display of a hierarchical menu from which one of the following selections can be made: how to integrate (Union, Associate, Generalise, Aggregate, Meet) and the direction of integration (from first subschema to second subschema, or vice versa).

SISIBIS helps in defining the 'how' and the direction based on the 'why' aspect as shown in Table 7.1. 'Union'ing is applicable to all equivalent types, while associating is applicable to all but disjoint entities. Aggregating and generalising are applicable to contained entities only, while 'meet'ing is only applicable to overlapping entities. For contained equivalencies, the direction of integration defaults to the containing entity so that it becomes the aggregating or generalising entity. In the case of establishing an association relationship between entities, the participant is prompted for the new relationship's name which is displayed directly below the integration direction icon. Similarly the participant is prompted for the name of the new generalising entity for the 'Meet' operator e.g. 'Student' in Figure 7.5.

The assertion specification is now complete and the participants submits it for deliberation by clicking the 'Save' button. SISIBIS then accesses the shared issue-base and updates it appropriately. If such a proposal or issue has already been submitted the submitter is notified, otherwise the status bar indicates a successful submission. In case the proposal was already submitted, the participant may go ahead by providing a statement of agreement to the proposal. The facilitator may remove a proposal via the 'Delete' button, after which the proposal is removed from the issue-base.

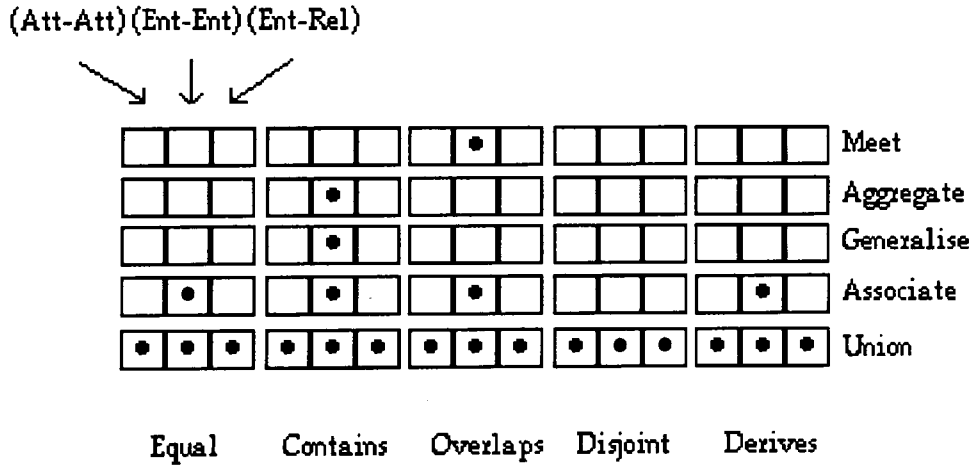

Table 7.1: Equivalence-Merge Type Matrix for Integrating Concepts.

5. The Deliberation Facility.

The deliberation facility is used to populate the issue-base with annotations. Deliberation takes place on a proposal or an issue that has been submitted. Deliberation starts with the participant exploring the integration stack and observing a proposal or issue (s)he agrees with or is suspicious of. Deliberations can be made on the 'why' or the 'how' of an integration. By pressing the mouse over the aspect a hierarchical pull-down menu is displayed showing the forms of deliberation available. The participant can choose one of the following deliberative contributions: explanation, question, answer, comment, agree, disagree and list participants who did one or any of them. For obvious reasons the submitter edits only two of the options: the explanation and the comment. Other participants may edit all options and in the case of an explanation, they are allowed to only read the explanation supplied by the submitter. An agreement or a disagreement is also annotated by an explanation. Hence an explanation may belong to an assertion, an agreement or a disagreement. Figure 7.6 shows such an explanation. Similar nodes are created for other forms of annotation except data scenarios which are discussed later in this section.

An annotation node is automatically linked to its originating proposal or issue on creation and the nodes are arranged in context by proposal or issue. The annotation is entered in free-text form and is also time- and date-stamped. An annotation is saved to the issue-base by clicking the 'Save' button, and deleted by clicking the 'Clear' button. The 'Microphone' button is used to send the entered annotation to specific participants currently logged onto the project. On invoking it SISIBIS scans the AppleTalk network selecting users who subscribed to the project. A dialogue box then appears on the screen with the list of participants from which a selection is made of those to which the annotation will be sent immediately.

Additional information in the form of a data scenario may be created by clicking the 'Data' button, resulting in the creation of a node into which an example related to the textual annotation is formally entered. Figure 7.7 shows such a data scenario exemplifying how a person's full name is derived from a concatenation of his first name and last name. The left half of the node contains instances of one concept while the right contains instances of the other. Corresponding instance values for the domain and range concept(s) of a mapping are entered as lists, one at a time by simply typing into the field. Though there is no validation of the entries, the editing process is 'formatted' in a context-sensitive manner: for each instance of the domain(s) concepts

there corresponds exactly one instance of the range concept(s). Data scenarios are linked, saved and deleted in the same way as annotations. However, since data scenarios are meant to augment an annotation, the link is to an annotation instead of a proposal or an issue. Browsing facilities are also offered to explore the various data scenarios given by the participants. Data scenarios are organised as nodes: a node per annotation per participant.

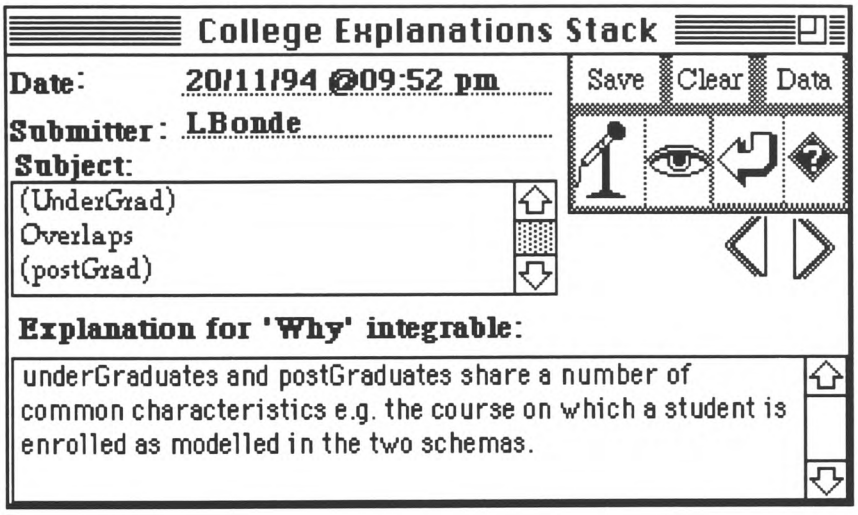

Figure 7.6: An Explanatory Annotation for an Assertion (proposal).

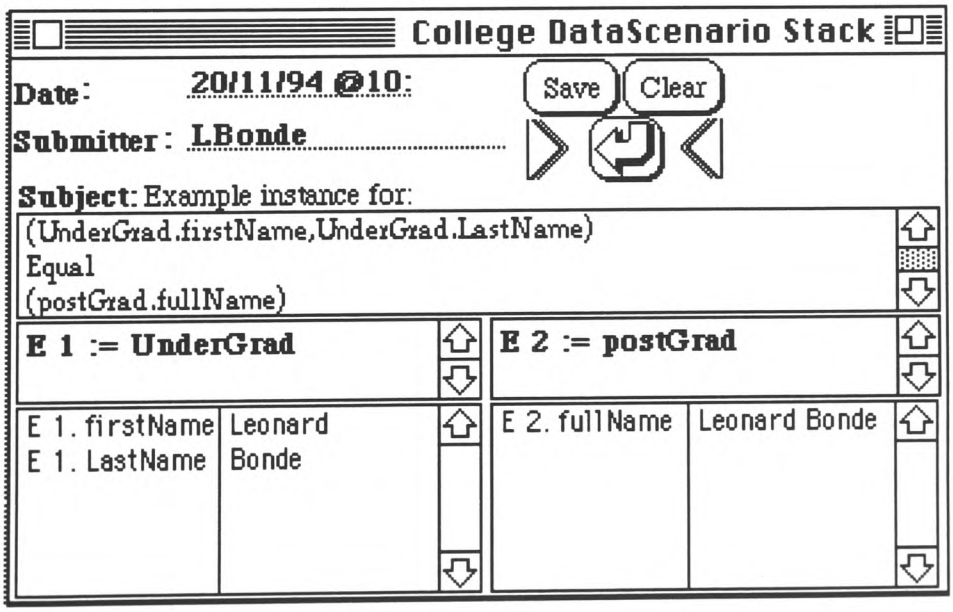

Figure 7.7: Data Scenario Exemplifying an Attribute Equivalence (issue).

Based on the annotations and data scenarios provided by the participants, the facilitator resolves the proposal or issue. Of all the proposals or issues provided for an integration, the DBA (facilitator) establishes links between them and the best integration proposal. This is achieved by establishing two sets of links: one to the solution node (proposal or issue) and another to the supporting or opposing (in case it is not a solution) arguments that led to the decision. If however, the arguments against integration are overwhelming, only one set of links is established: that to the opposing arguments that influenced the decision. The links are achieved via a combination of browsing and a 'link to this argument' floating palette that appears on the screen (c.f. Figure 7.9). The links effectively establish the rationale for the integration decision by defining a decision-trail that can be traced later on.

6. The Merging Facility.

On the instruction of the DBA, SISIBIS attempts to automatically merge the concepts involved in the proposal. No merging takes place until the proposal is resolved or all the proposal's issues are resolved. The process of merging then checks the 'how'-integrate aspect of a proposal together with the direction of the integration. It first constructs a template representing the resultant entity or relationship. It then appropriately enriches the resultant entity or relationship by allocating equivalenced attributes to it. For relationships, the member entities are updated to reflect any previous mergers involving them.

It then checks if there are any remaining attributes in the other entity or relationship, if not then the amalgamation is considered a transformation (e.g. if no attributes remain for an entity then the entity is considered to have been decomposed). Based on the accepted key equivalence, SISIBIS assigns the appropriate cardinalities to any new relationships introduced using guidelines provided by [Batini et al 1983]. This is done by translating the constraints that exist between key attributes and non-key attributes i.e. there exists one-many relationship between the key and the other attributes. For instance, if the attribute was a key in its original residing entity/relationship, then the entity derived from it is always on the one-side of the created relationship, otherwise it belongs to the many side. Optionalities of such relationships initially default to mandatory, but may be modified on visiting the evolving schema. SISIBIS then visits the subschema entries of the concepts and collects their information such as attribute descriptions and properties. The template is then stored as a node in the entity or relationship dictionary of the evolving schema.

Finally, SISIBIS establishes a link from the nodes of the original concepts of the subschemas to the new concept node, and another from the new concept node to the first node of the decision-chain (rationale) in the issue-base. Figure 7.8 illustrates this. Traversals may then be made from the subschema concepts to the integrated concept in the evolving subschema. Such a traversal is tantamount to tracing a mapping but it must not be taken to faithfully represent a mapping. For instance, the trace from 'FirstName' to 'Name' is into 'Name', hence not a faithful representation of the mapping. It however gives a feel of the derivation relationship between the attributes in question.

Another facility serving a related purpose appends all unintegrated concepts to the evolving schema resulting in a partial integrated schema. All relationships are appended after participating entities have been appropriately updated to reflect the implemented integrations.

SISIBIS uses a linking mechanism based on tagging a node with the unique system-generated identification number of an anchor node. It uses this mechanism to link annotation nodes to an integration node as well as linking all alternative proposals to a resolution node. The same technique is used for linking the rationale list to a proposal or issue node.

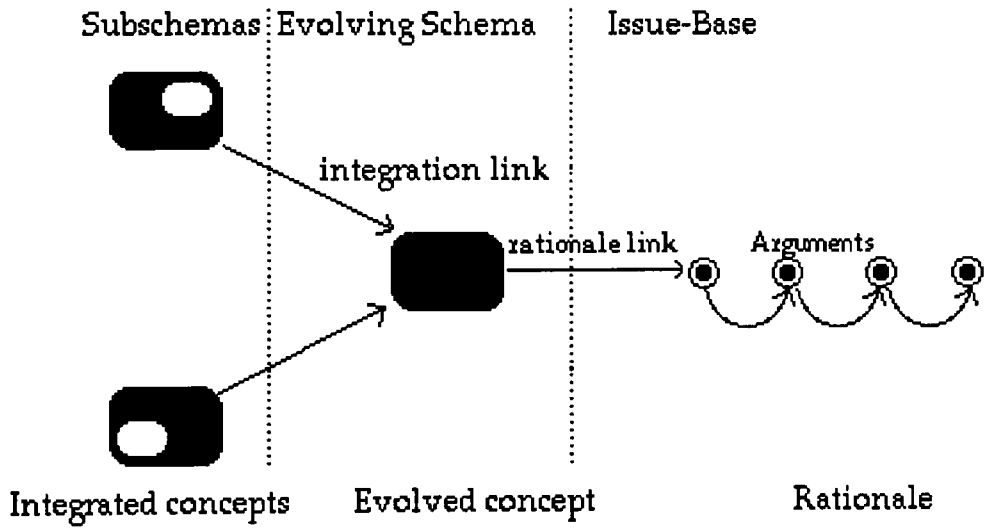

Figure 7.8: Results of a SISIBIS Operation.

7. SISIBIS Analytical Support.

Though it was initially intended to include extensive analytical support in SISIBIS, only simple analysis is possible at present. These include an analysis of:

a. the naming conflicts of homonyms based on the accepted equivalencies.
b. checking whether the semantics of key equivalencies echo entity/attribute equivalencies.
c. checks for any unresolved proposals or issues and any unintegrated proposals.
d. in conjunction with the analytical facilities of HSMS (c.f. Figure 4.1) proposals that may be implied by a validated proposal may be suggested by the demonstrator. After having appended all unintegrated concepts in the subschemas to the evolving schema, an analysis may suggest entities or relationships that seem to be capturing the same semantics. In case there are any, the user notes them, reverts back to the integration dictionary and submits another proposal, thus triggering further deliberation. For instance, having validated the integrability of two pairs of entities the possibility of integrating the relationships in which the entities are members may be highlighted.

8. Browsing.

Equivalence/conflict analysis is aided by browsing the dictionaries. The relevant information to determine which nodes to visit may be derived from the subschemas or the evolving schema. There are two gateways to the subschemas. The first option is to go to the entity-relationship dictionary of a chosen subschema and glean semantic information. This is done by clicking the mouse on the schema name displayed on the card. The participant is promptly taken to a separate window from which browsing is re-enacted by the browsing facilities offered in HSMS. For instance, this may be the approach taken by a person who is looking for an equivalence or a conflict.

The other option which may be taken by a respondent to a proposal or issue is to go directly to the node containing the concept or to an integration node that involves a selected concept. This is achieved by pressing the mouse over the concept field. A hierarchical pull-down menu then appears giving the participant an option to view (a) a concept in its subschema, (b) an integrated concept in the evolving schema, or (c) a list of related assertions involving the concept. In either case, the participant is taken to the appropriate node. Browsing tools are also offered to go to the next or previous proposal or issue, to see other proposals/issues involving either of the current concepts, and to view the resolution. A resolution node displays the resolution and forms the source of a decision-trail linking-up the annotations that led to the resolution. Another form of browsing requests that, for a given participant, the system

provides all the proposals or issues submitted by him/her from which one is selected and visited.

Browsing the issue-base is done via two basic browsing facilities. Firstly, a simple facility that allows one to go to the next or previous annotation is provided. The traversals in this case are context sensitive i.e. limited to the context corresponding to the original proposal or issue. Secondly, there is the more elaborate 'see also' button that allows the selective traversals by participant name. This presents a dialogue box listing all the participants, and on selecting one, another dialogue box appears listing all the annotations entered by the participant.

Equally important are the browsing mechanisms employed during the processes of resolution (and rationale) definition (and touring). These browsing mechanisms utilise a floating palette of tools. This was done in preference to a dialogue box to reduce overlaying integration data and the inconvenience that is associated with switching between the integration window and the dialogue box after a traversal is executed. Figure 7.9 shows these palettes.

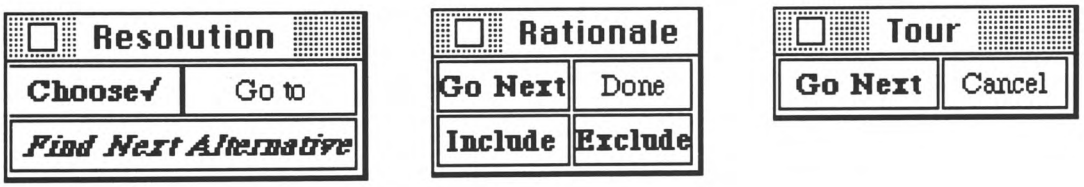

Figure 7.9: Resolution, Rationale and Tour Browsing Palettes Respectively.

The 'Resolution' palette offers three facilities:

a. the 'Choose' button to select the current assertion as the choice amongst all the assertions involving the concepts in question.
b. the 'Go to' button to traverse to the assertion selected as the resolution.
c. the 'Find Next Alternative' button that takes the reader to the next alternative assertion. Thus the DBA browses only those assertions related to the current one.

The 'Rationale' definition palette offers the reader four facilities:

a. the 'Go Next' button that takes the reader to the next annotation node for an assertion. The reader can then 'Include' or 'Exclude' it.

b. the 'Done' button that terminates the rationale definition process and links the rationale list to the appropriate assertion.
c. the 'Include' button that appends the current annotation node to the end of the rationale list of the resolution assertion.
d. the 'Exclude' button that removes the current annotation from the rationale list of the resolution assertion.

The 'Tour' palette offers the reader a guided tour of the rationale list defined by the DBA. It offers two facilities: one to traverse to the 'Next' annotation in the rationale list and the other to 'Cancel' the tour.

9. Collaborative Schema Integration Example Using SISIBIS.

The example that follows uses subschemas shown in Figure 7.10 (extracted from [Navathe et al 1986]). The order of integration is Figure 7.11(a) - Figure 7.16.

However, a different order of integration can be employed but the number of stages and indeed the resultant integrated schema may differ. Figure 7.11(a) and 7.11(b) illustrate assertions involving the entities 'topic' and 'keyword' and the attributes of these entities respectively. At this stage, our binary integration tree consists of two nodes (Figure 7.11(a), Figure 7.11(b)) and rooted at Figure 7.11(a). The participant (who happens to be the DBA in this case) asserts that the two entities (attributes) are equal because they refer to the same real-world concept. The participant intends to resolve this duplication by uniting the concepts and retaining the names used in the first concept. After deliberating, reaching consensus and choosing this assertion the implementation adds the entity 'Topic' to the evolving schema's entity dictionary and discards the entity 'Keyword'. Similarly, the attribute 'name' is selected in preference to 'title'. The integration of the relationships shown in Figures 7.14 and 7.15 are implemented in the same way.

Figures 7.12(a) and 7.12(b) illustrate a different assertion involving the entities 'Publisher' and 'Publication'. The participant asserts the entities overlap (Figure 7.12(a)) because they have attributes that represent the same real-world concept: a publisher (Figure 7.12(b)). The participant intends to resolve this duplication by creating an association relationship called 'publishes' between the entities. On choosing this assertion after deliberation, the implementation adds these two entities and the new relationship to the evolving schema's dictionaries. The attribute 'name' is retained in the entity 'Publisher', while the attribute 'publisher' is discarded from the entity

'Publication' since 'name' is a key attribute of 'Publisher' (not shown in the illustration).

Figures 7.13(a) and 7.13(b) illustrate the assertions that 'Publication' generalises 'Book' (i.e. all books are publications) and that the identifying codes for both concepts is the same, respectively. An implementation of the proposal enters the two entities and the generalisation relationship (depicted in Figure 7.16) into the evolving schema's dictionaries. In addition, the attribute 'Code' is retained in the entity 'Publication' and is inherited by the entity 'Book' through the generalisation relationship. Because the entity 'Publication' has been involved in a previous assertion (Figure 7.12), this integration starts a new stage of the binary-integration strategy used.

Finally, Figure 7.17 shows the E-R diagram of the resultant conceptual schema drawn using the diagramming tool offered in HSMS.

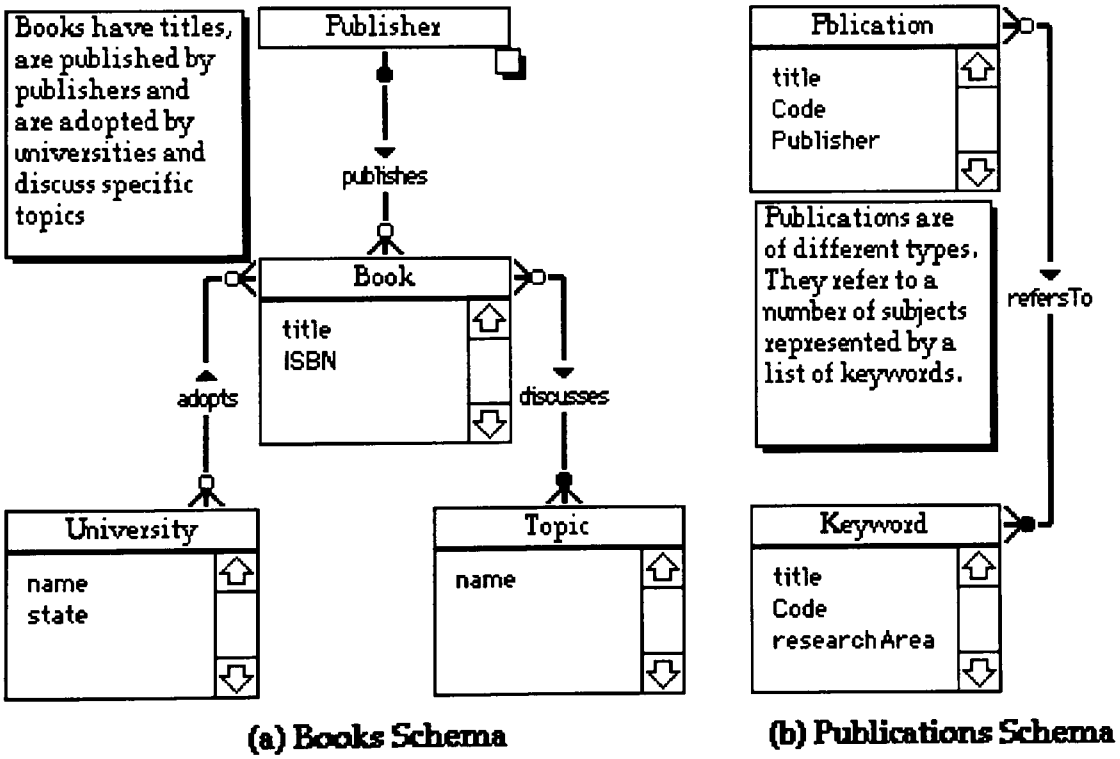

Figure 7.10: Sample Subschemas for Integration.

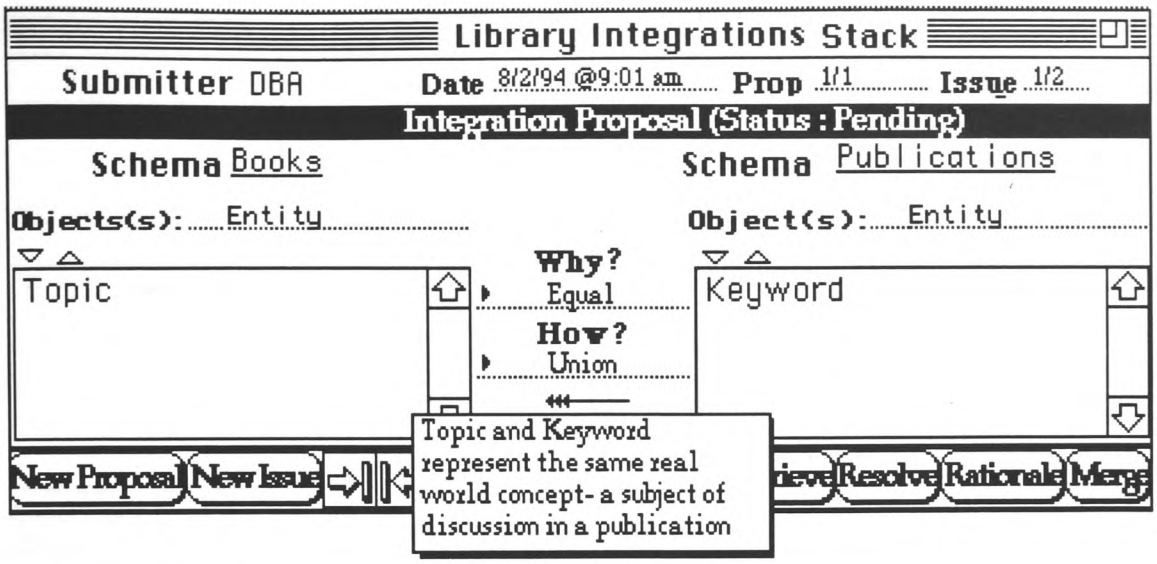

Figure 7.11(a): Pending Entity Integration Assertion.

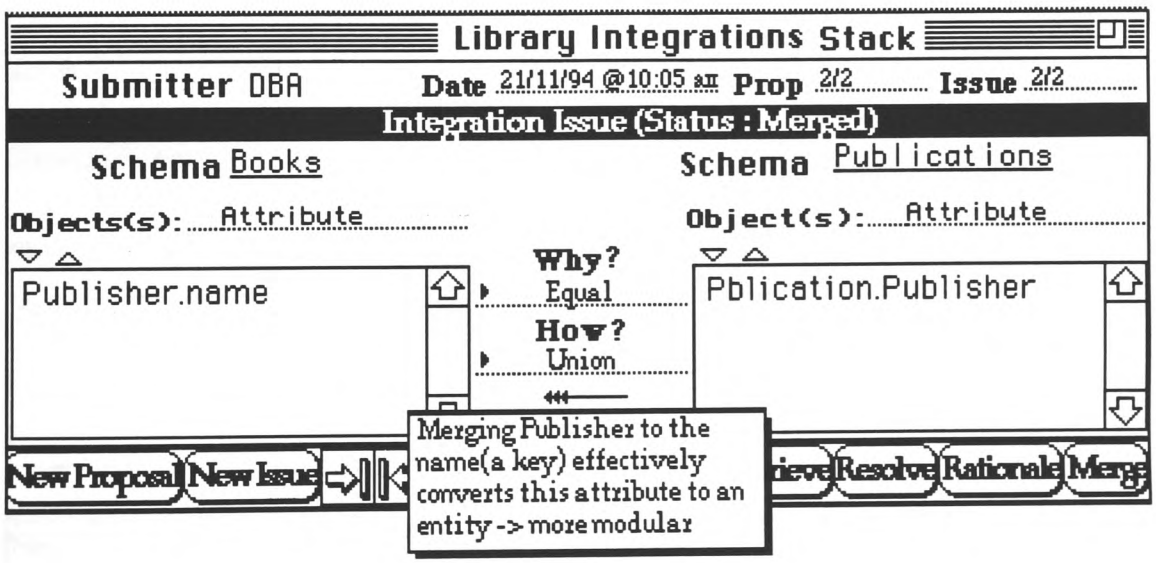

Figure 7.11(b): Submitted Attribute Integration Assertion.

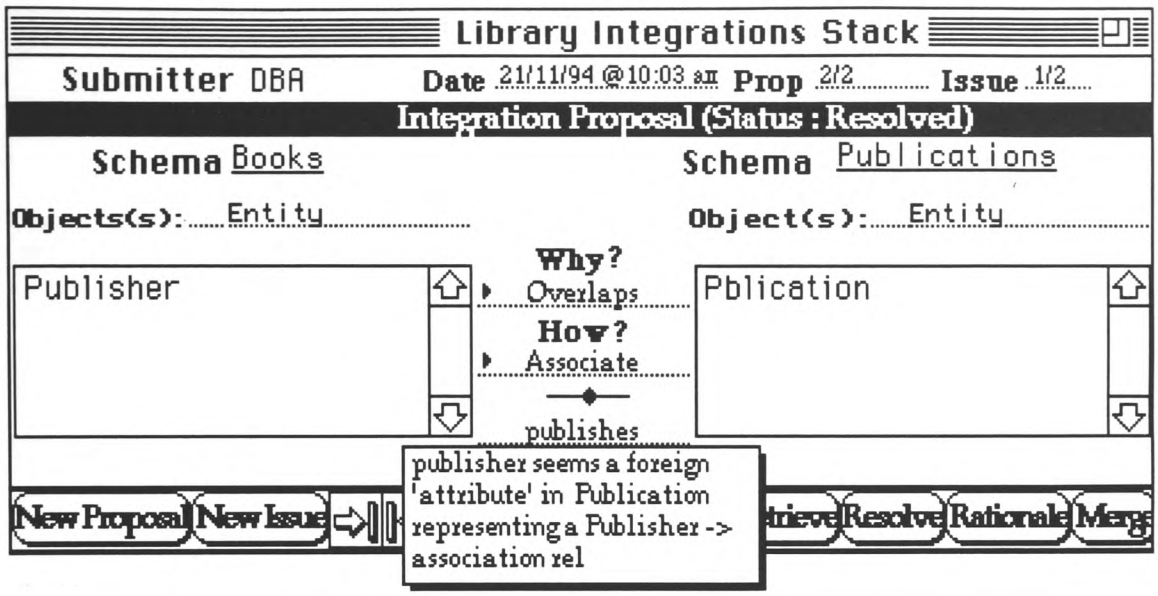

Figure 7.12(a): Resolved Entity Integration Assertion.

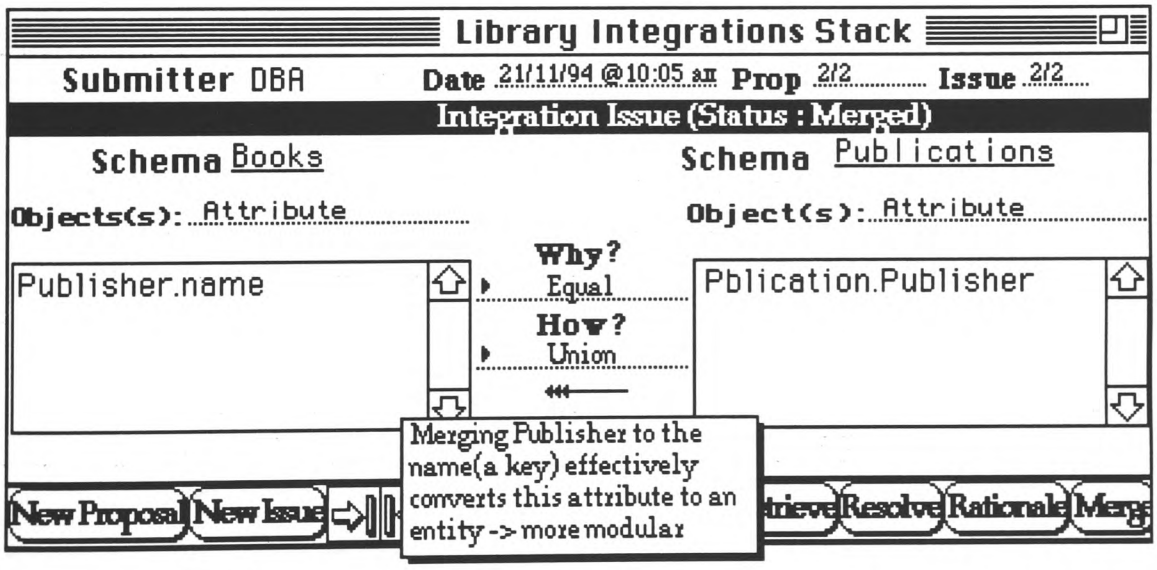

Figure 7.12(b): Merged Attribute Integration Assertion.

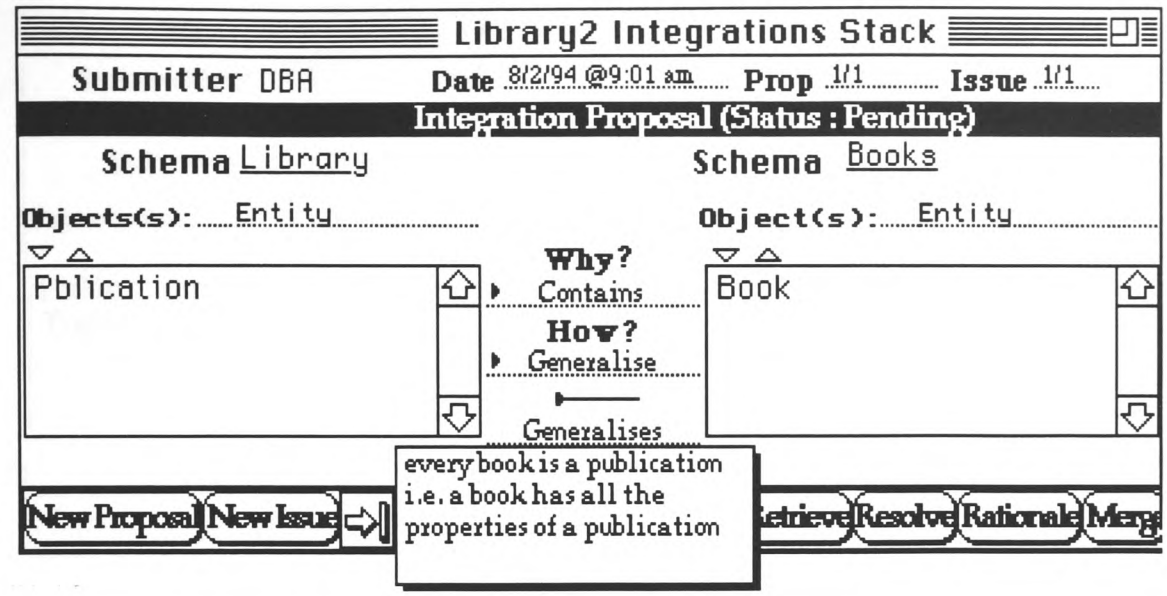

Figure 7.13(a): Pending Entity Integration Assertion.

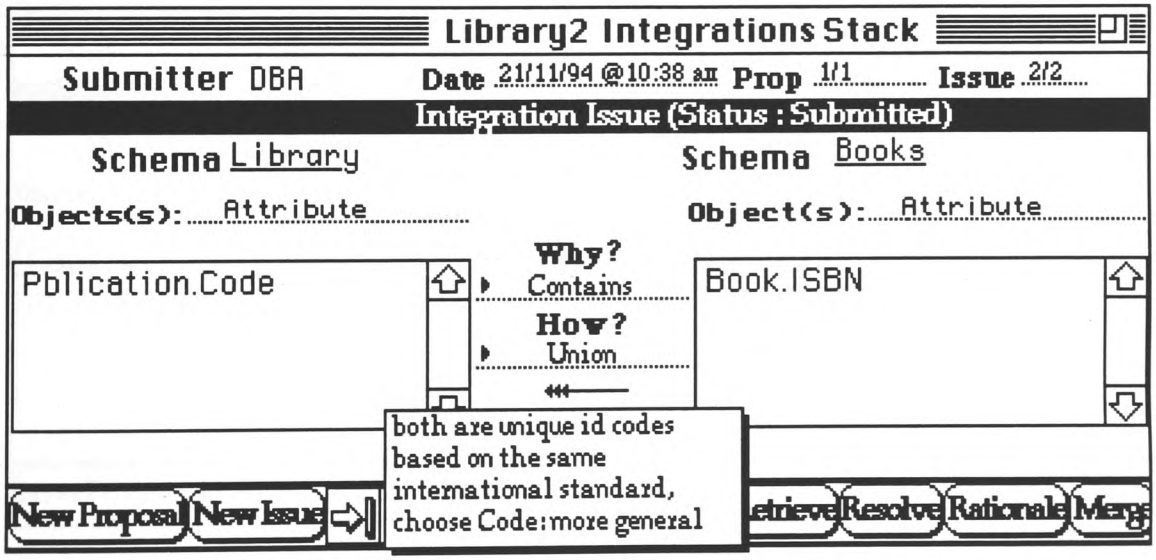

Figure 7.13(b): Submitted Attribute Integration Assertion.

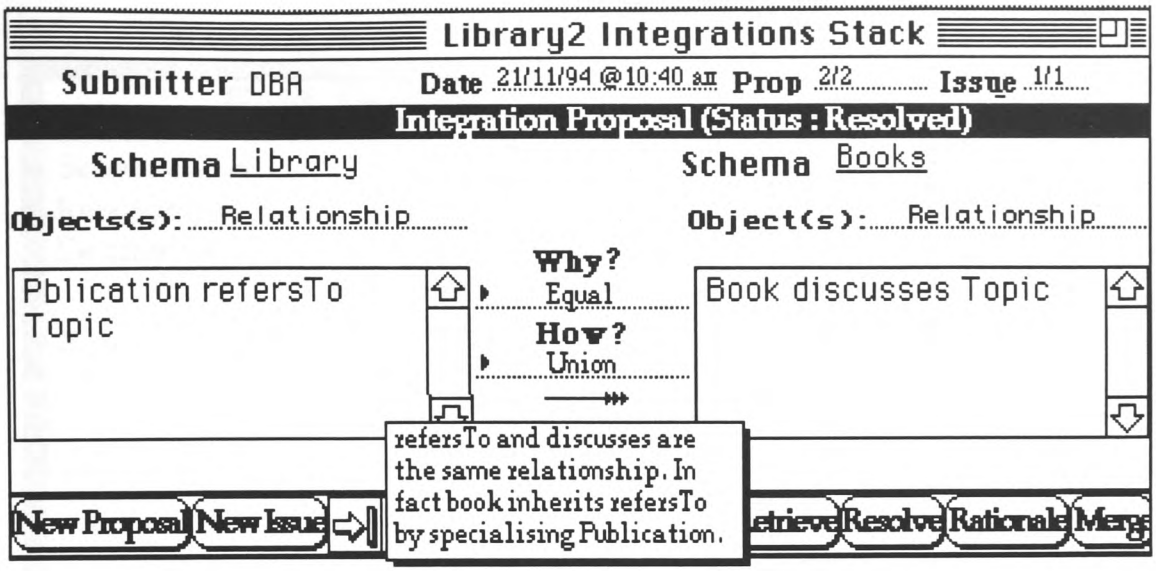

Figure 7.14: Resolved Relationship Integration Assertion.

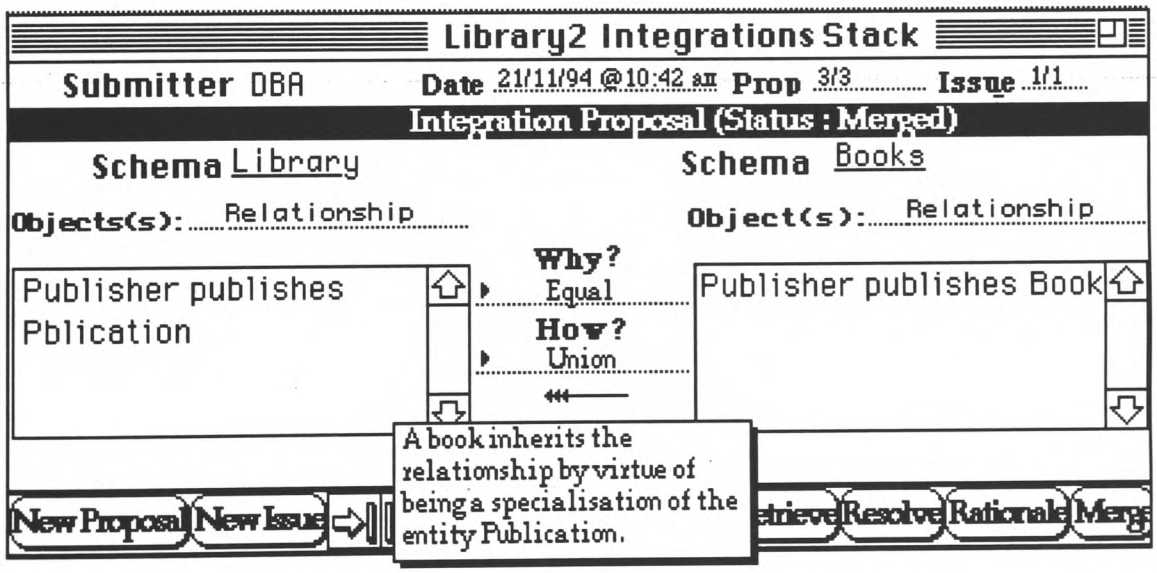

Figure 7.15: Merged Relationship Integration Assertion.

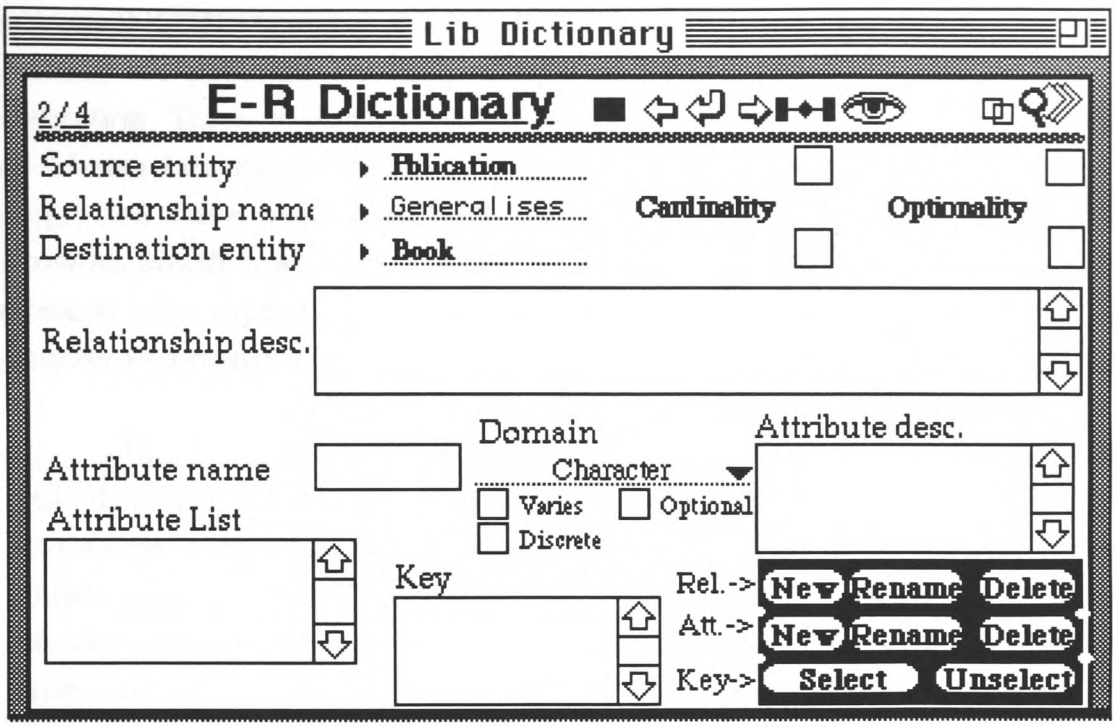

Figure 7.16: Implementation of the Resolved Assertion of Figure 7.13(a).

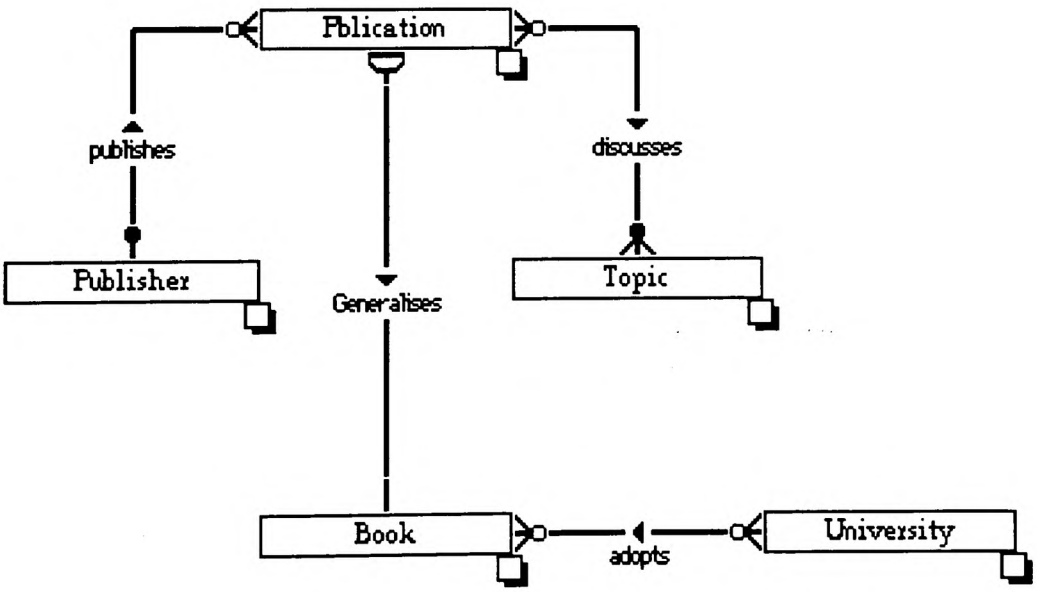

Figure 7.17: ERD of Conceptual Schema Resulting from Integrations of Figures 7.11(a) - 7.15.

10. Related Work.

The work discussed in this chapter is related to a number of other research

projects. INCOD [Atzeni et al 1982], IMT [Lundberg 1982] and DDEW [Reiner et al 1984] are some of the early CAISE tools developed to address schema integration. These systems however focus mainly on document management for schema integration. Lately a number of CAISE tools incorporating schema integration have been discussed in the literature. They vary from using heuristics [Sheth et al 1988] to knowledge-based systems [Collet et al 1991] and to multi-user support for schema integration [Hayne and Ram 1990]. We compare and contrast SISIBIS with these tools in the following subsections.

a. 'A Tool for Integrating Conceptual Schemas and User Views' developed by [Sheth et al 1988]. This tool is probably the closest to SISIBIS. It uses an extension of the E-R model called the ECR (Entity-Category-Relationship] model [Elmasri 1985]. Basically, the tool utilises an n-ary strategy with an iterative-binary approach to concept integration. The tool collects schemas and integration assertions via a series of forms. It presents the DBA with an ordered list of paired concepts based on an 'attribute ratio' which is used as a measure of similarity. The tool however assumes that all naming conflicts have been resolved. Unlike SISIBIS, the tool does not attempt to capture the rationale of integration or to support deliberation. While employing naming standards during schema modelling might go a long way to reduce ambiguity, we argue that problems in the interpretation of names still persist. Another major difference is that SISIBIS offers flexible navigation across the integration dictionary, the input schemas and the evolving schema. Further SISIBIS provides support for integration semantics elicitation. Like SISIBIS, the tool also implements concept integration in the evolving schema.
b. The MUVIS (Multi-User View Integration System) project [Hayne and ram 1990]. MUVIS is similar to SISIBIS in that it uses the EERM, it is multi-user and it has co-operating subsystems:

   i. the View Modelling System (VMS which serves the same purpose as HSMS).
   ii. the View Integration System (VIS which serves the same purpose as SISIBIS).

However its approach is fundamentally different. It is a knowledge-based system using an expert system to compare concepts in a binary fashion and

automatically computes assertions about concept equivalencies as well as integrating the concepts. MUVIS is also fundamentally different in that it allows the database designers to directly enter the concepts into the shared schema. Thus it does not seem to have a subschema selection stage. Similar to the tool described in (a) above, MUVIS also assumes the existence of a synonym or homonym lexicon. Unlike SISIBIS, its preintegration phase does not generate or capture the semantics for integration. MUVIS's calculation of the probability of equivalence between concepts seem to have a sound theoretical basis. MUVIS goes further in implementing integrations by supporting abstraction mechanisms in relationship integration. MUVIS also supports mapping between the subschema and the integrated conceptual schema. However, it does not support collaboration thus failing to capture the semantics and the rationale of integration. Thus it is difficult to know how certain integration decisions were arrived at.

c. **Cyc-Carnot Project** [Collet et al 1991]. The Cyc-Carnot project at MCTC (Microelectronics and Computer Technology Corporation) uses an existing global schema known as the Cyc knowledge-base [Lenat and Guha 1990]. Instead of merging subschemas to each other, the tool merges the subschemas individually to the Cyc knowledge-base. The Cyc knowledge-base is claimed to have enough entities and relationships to cover the requirements of most information areas!. The distinguishing factor about the Cyc-Carnot project is that in addition to utilising schema knowledge it also utilises corporate rules. Almost on the same lines as SISIBIS, the Cyc-Carnot project utilises information elicited from designers (e.g. comments, guidance from the integrator) and knowledge about the data model. It uses what are termed 'articulation axioms' for schema representation and concept matching. The tool uses knowledge-based techniques (frames) with extensive formalisation to achieve this. However it does not seem to support collaboration (at least between the designers and the integrator) and navigation. Another positive factor about the project is mapping between subschemas and the global conceptual schema resulting from the integration.

11. Conclusion.

This chapter has described the functionality of a prototype system for collaborative schema integration, SISIBIS. We reiterated the problems of schema integration and highlighted the shortcomings of automated schema integration. The prototype described adopts a human-intensive approach in

which the integrators drive the process by providing assertions for concept integration and the integration method for implementing the integration. The prototype assists the integrators with document management, browsing the subschemas, some analytical support and implementing the mergers. In addition to providing assertions, the integrators deliberate over the assertions by annotating them with their opinions of the assertions. The annotation may take the form of an explanation, a comment, a question and an answer, as well as an exemplifying or counter-exemplifying data scenario. Such deliberation serves three vital purposes:

a. intelligence gathering to establish semantic consensus.
b. the establishment of the rationale for an integration decision.
c. knowledge elicitation culminating in the establishment of an integration knowledge-base that may be referenced during maintenance of the resultant database and possibly the identification of areas of duplicated functionality (and data) within an organisation.

We also described how the prototype aids in 'effectively' eliciting data semantics and organising the knowledge in a database structure based on an extension of Rittel's IBIS scheme. The IBIS scheme [Rittel and Kunz 1970] and hypermedia have jointly been proposed for collaborative work [Sihto 1989, Irish and Trigg 1989, Conklin and Begeman 1988, Fischer et al 1992, McCall 1991]. The IBIS scheme attempts to structure the process of deliberation by categorising deliberation artifacts into issues (the problems to solve), positions (the individual solutions to the problem), arguments (statements explaining the solutions) and resolutions (the agreed upon solutions). This scheme is appropriate for collaborative schema integration because it supports the construction, comparison and evaluation of solutions to an integration plan. Hypermedia [Conklin 1987, Rada 1991] is an important environment for schema integration because it supports unstructured databases of information. Firstly, the linear/non-linear nature of hypermedia reduces the cognitive overhead by presenting concise information embedded with links (for selective access), thereby making the learning process easier. Secondly, hypermedia's unstructuredness blends well with the aspects of annotation, namely: explaining, commenting, questioning, answering and exemplifying. Thirdly, hypermedia enhances these benefits by providing an environment to electronically maintain the integration models and the informally generated information in hypermedia nodes.

Integration proposals are processed using 'propose-resolve-merge' cycles. Further, we described how SISIBIS is used to define the rationale of an integration decision by linking up the annotations provided by stakeholders.

Also described is how the documentation is organised into self-contained units. The discussion illustrated how hypermedia has been utilised for integration dictionary authoring, browsing, aggregation and annotation. We showed how the hypermedia artifacts of nodes and links are employed as organising themes in the integration dictionary. Foremost is the adoption of the binary strategy for schema integration which naturally enforces modularity. Such modularity reduces the information overhead required for integrators, thereby making the integration process more manageable. This contrasts sharply with an n-ary strategy in which it is difficult or even impossible to fit all the involved concepts on one page. Also described was how an evolving schema iteratively develops into a partial integrated schema and how the partial integrated schema iteratively develops into the final conceptual schema.

Finally we compared and contrasted SISIBIS with other related tools. The major difference between SISIBIS and the tools discussed in the literature is the collaborative dimension added to schema integration in SISIBIS. The next chapter discusses an evaluation of SISIBIS, and some feedback on the schema modelling subsystem, HSMS.

Evaluation Of The Demonstrator.

Summary.

The demonstrators discussed in this thesis, HSMS (cf. Chapter 4) and SISIBIS (cf. Chapter 6), were developed to demonstrate the features of a hypermedia-based CAISE tool for the conceptual modelling process of database design. HSMS and SISIBIS together form a hybrid hypermedia system developed using HyperCard and Pascal. This chapter discusses an evaluation which was carried out to assess the usefulness of the demonstrators. The author was given funding by the University of Glamorgan to conduct a formal evaluation of the demonstrators over a period of 3 months. For the purposes of this thesis, a fundamental decision was taken to concentrate on an evaluation of SISIBIS: the collaborative schema integration facility. It is particularly in this area that we feel a contribution to knowledge has been made. A qualitative evaluation of the demonstrators has also been conducted both prior to and during the formal evaluation. Because of the nature of the project and the constraints placed upon it, both types of evaluation are necessarily preliminary. However, a number of suggestions as to future evaluation work are discussed in section 9.

Section 1 introduces the chapter by outlining some issues relevant to evaluating computing systems, and discusses the aim of the evaluation in the context of these issues. Section 2 describes the evaluation plan in detail, highlighting the evaluation parameters and the laboratory set-up for the evaluation. Section 3 is devoted to the important issue of sampling evaluation subjects and setting evaluation tasks for the evaluation. The evaluation sessions and their results are discussed in sections 4 and 5 respectively. Section 6 supplements the formal evaluation by describing the informal evaluation conducted during development of the demonstrator. It also discusses an evaluation done over a longer period of time and with more complex subschemas than those used for the formal evaluation. Some usability issues arising from the evaluation are itemised in section 7. Section 8 outlines a number of the major problems experienced during the course of the evaluation. Section 9 describes what we consider to be an ideal evaluation of the work. Finally, section 10 summarises the results of the evaluation.

1. Introduction.

Evaluating all aspects of diverse environments such as computer-supported collaborative systems (e.g. SISIBIS) is practically impossible. Computer-supported collaboration is a very broad subject touching on

economic, technical and social aspects of work arrangements and practices in natural settings [Olson and Olson 1991, Applegate 1991, Grudin 1990, Wagner 1993]. The natural setting for our demonstrator would ideally be an information systems department engaged in large-scale database development work. However, it has proven difficult to get such 'natural' subjects to participate in an evaluation of the work because of time and resource constraints. Therefore, we have chosen to evaluate the demonstrator using subjects in an academic setting with various levels of experience of database development. In such a setting economic and social aspects are virtually impossible to assess. Hence we focused on evaluating the technical aspects solely.

1.1. Assessing the usefulness of computing demonstrators:

[Nielsen 1993] provides a framework for evaluating the acceptability of computing systems (Figure 8.1). He maintains that two types of acceptability can be distinguished: social acceptability and practical acceptability. Social acceptability is concerned with how a system impinges on the work practices and arrangements of its intended users. Practical acceptability concerns issues of incorporating a system into the current work setting. Since our demonstrator was never intended to be a fully-fledged system, our evaluation was limited to the issue of practical acceptability.

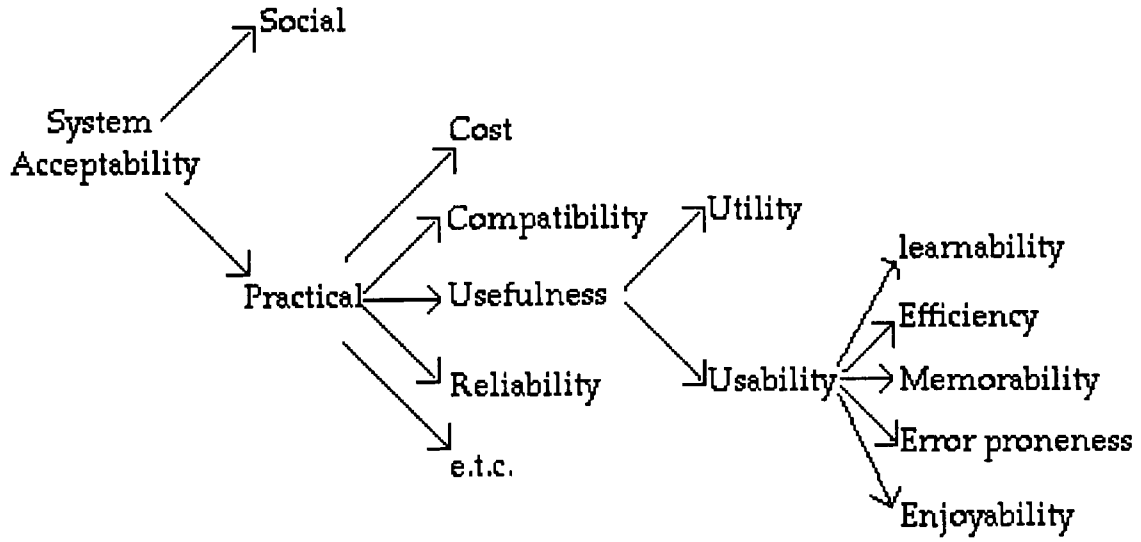

Figure 8.1: Model of Aspects of System Acceptability [Nielsen 1993].

The practical acceptability of a system involves a number of factors such as cost, compatibility, reliability, usefulness etc. Because our system is designed as

a research demonstrator we only addressed the issue of usefulness. [Nielsen 1993] further distinguishes between two aspects of usefulness: utility and usability. Utility concerns the question of whether the functionality of a system in principle does what is needed (i.e. does it offer the right functionality). Usability concerns the question of how well users can use the functionality of a system. Our evaluation is targeted at the issue of utility. Since our demonstrator is not a fully-fledged system, we acknowledge that the demonstrator is deficient in a number of usability areas. Some of these usability problems are discussed in section 7.

However, it is difficult to separate out issues of utility from issues of usability because, by its very nature, a computing system is only ever used through its interface. The experience-level of users in relation to a system also affects this relationship. One would expect, for instance, inexperienced users to be more aware of (responsive to) usability issues than experienced users because of the learning curve naturally associated with system use. Realistically, the separate evaluation of usability and utility is to a large extent dependent on the level of expertise of the subjects. For instance, [Twidale 1993] cites *'the effect of the user-interface'* as one of the disadvantages of a controlled evaluation. Evaluating utility requires experienced subjects who can objectively manipulate the interface to get the system to reveal its functionality. Ideally, utility-targeted evaluations require that inexperienced subjects need training first.

1.2. Aim of Evaluation.

The objective was to evaluate SISIBIS, touching upon key features of HSMS that are relevant to schema integration. The specific aim of the evaluation was to test the hypothesis that SISIBIS (as a hypermedia problem-resolution system) lends itself to supporting:

a. Collaboration: we intended to establish whether the tool supports the collaborative nature of schema integration i.e. whether:

   i. it facilitates explorative schema integration problem solving.
   ii. the link node architecture of hypermedia aids deliberation.

b. Complexity: we intended to establish whether SISIBIS (and HSMS where relevant) aids in managing the complex inter-relationships of database design documents.

A classic experimental evaluation needs operationalisation in terms of both independent variables (the predictors) and dependent variables (the effects). Our independent and dependent variables are shown in Table 8.1. Rows 1 and 2 of the table have been used as a test of SISIBIS's support for collaboration, while row 3 tests whether the structural mechanisms of SISIBIS aid in managing the complex inter-relationships among database design documents. Details of how the independent variables were varied, how data for the dependent variables was collected, and predictions for the collected data (hypothesis) is discussed in section 2 below.

	Independent Variable for Evaluation	Dependent Variable for Evaluation
1	Task size	task completion time
2	Availability of issue-base	task completion time. quality of integrated schema. degree of conflict. degree of annotation.
3	Availability of links to nodes	task completion time usefulness of traversal

Table 8.1: Independent and Dependent Variables Used to Evaluate SISIBIS.

2. Evaluation Plan.

In this section we discuss how the evaluation was operationalised in terms of: parameters (the independent and dependent variables), evaluation tasks, evaluation subjects and the laboratory set-up. We then discuss how the sessions were conducted in terms of data capture, data analysis and the hypothesis they tested.

The overall evaluation plan consisted of assigning the same integration tasks of increasing size to different groups of subjects. Different groups enjoyed different facilities of the demonstrator. Typically, the difference was in the presence or absence of: the issue-base or links. We wanted to determine how these independent variables (task size and availability of facilities) affect the following dependent variables:

a. the time it took to complete each of the integration tasks.
b. the 'quality' of the final integrated schema as measured by the complete representation of the semantics embedded in the subschemas. A schema

with the least redundancies and that embedded most of the semantics in the subschemas was considered more representative.

c. the degree of conflict among assertions, and

d. the degree of annotation of assertions.

During the session, the author observed and took notes about the progress and problems of each session. At the end of the session the subjects were given a questionnaire (shown in Appendix 1) so as to glean information about the subject's knowledge of schema modelling and integration, as well as any feedback on aspects of the demonstrator that are difficult to evaluate objectively (such as subjective satisfaction, possible anxieties).

In summary, this design helped us to assess the utility of SISIBIS:

a. within-subjects (i.e. for a specific group with given SISIBIS facilities at its disposal) for the different tasks of varying size.

b. between-subjects (i.e. between groups with specific but different SISIBIS facilities at their disposal) for corresponding tasks of varying size.

2.1. Evaluation Parameters (variables).

In general, the major independent variables identified to test the hypothesis were the availability or inavailability of: nodes, links and annotations of the issue base. However, since nodes form the bedrock of links, it was impossible to vary the nodes in a controlled and systematic manner viz. a viz. granularity, contexts and organisation. This restriction is fundamentally due to the much publicised problem of fixed-link hypermedia architectures [Alschuler 1989, Bieber 1992]. The intimate connection between the structure, content and presentation of our design hyperdocuments made it difficult to alter the structure of these hyperdocuments later. Figure 8.2 shows how variation of the remaining parameters, links (whose presence is indicated by the presence of arrows in the figure) and annotations, was effected:

a. Links.

There are a number of links in the demonstrator. However, the major links varied are from the integration dictionary to the:

i. subschemas, and

ii. annotations in the issue-base.

These were varied by enabling them for some groups (Figure 8.2(a) and (c)) and disabling them for the other groups (Figures 8.2(b) and (d)).

b. Annotations.

Some groups enjoyed the annotation facilities of the issue base (Figure 8.2(c) and (d)), while others did not (Figure 8.2(a) and (b)).

Comparing and contrasting the data collected for sessions corresponding to:

a. figures 8.2(a) with 8.2(c), and figures 8.2(b) with 8.2(d) provided an opportunity to assess the effect of deliberation on schema integration. This was used to test the hypothesis for collaboration.
b. figures 8.2(a) with 8.2(b), and figures 8.2(c) with 8.2(d) provided an opportunity to assess the effect of links on managing schema integration documents. This tests the complexity hypothesis.

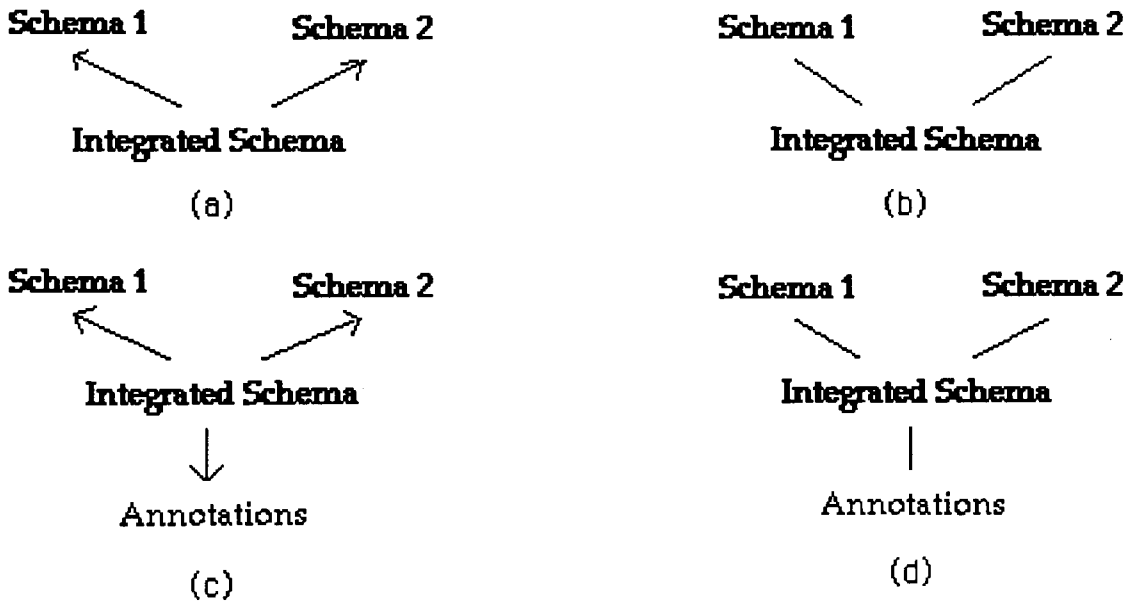

Figure 8.2: Evaluation Combinations for Links and Annotations.

2.2. Laboratory Set-up.

The initial plan for the evaluation was to make all the groups work on the tasks simultaneously. During a pilot run under such a set-up, the performance of the demonstrator degraded due to the presence of many subjects on the network. Also the pilot run quickly showed that difficulties would be encountered with respect to making observations and taking notes. Further, the subjects would initially need the experimenter's undivided

attention to give them confidence (as the majority of them had little schema integration knowledge and little prior exposure to SISIBIS). It was therefore decided to scale-down the sessions to only one group per session.

The laboratory set-up took two forms (Figure 8.3) corresponding to the groups that documented their deliberations (Figures 8.2(c) and (d)) and those that did not (Figures 8.2(a) and (b)). All groups had SISIBIS running on computer under the following set-ups:

a. a multi-workstation set-up comprising three computers: a server and two clients. The facilitator created integration schemas on the server machine, while the other members of the group worked on the client machines. The subjects were discouraged from having direct verbal communication except with the facilitator. They could however communicate via the messaging facilities offered by the demonstrator.
b. a single workstation set-up comprising one machine and three subjects working on it. The subjects had direct verbal communication among themselves.

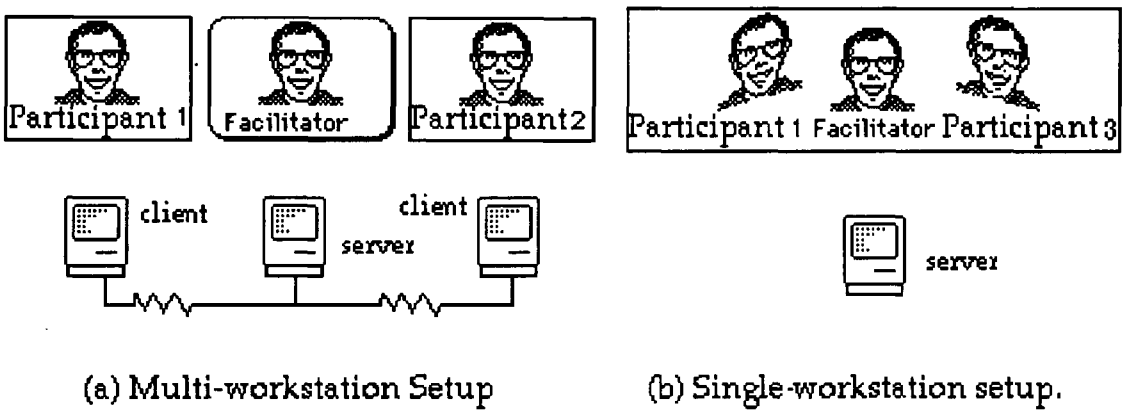

Figure 8.3: Laboratory Setup for Evaluation Sessions.

3. Sampling of Evaluation Subjects and Tasks.

Sampling is an integral part of any evaluation and involves selecting the evaluation tasks as well as the subjects that will perform the selected tasks [Nielsen 1993]. Both the tasks and subjects should be as representative as possible of the setting to which the computer system will be placed. The tasks selected are however dependent on the ability of the chosen subjects. Sections 3.1 and 3.2 discuss how these aspects were selected for our evaluation.

3.1. Evaluation Subjects.

[Nielsen 1993] states, as a basic rule for sampling evaluation subjects, that subjects should be as representative as possible of the intended users of the system. Three types of user expertise can be identified: computer, domain and system (Figure 8.4). Computer experts are users who are generally comfortable with working with computers. Domain experts are users who are expert in the application area of a system. For instance, our domain experts would be data modellers engaged in modelling the subschemas to be integrated. System experts are users with experience of using a given system.

Industrial database design practitioners would ideally have been worthy subjects as they are more likely to possess the first two qualities. However getting them to invest in such an evaluation has proven difficult. Consequently, the evaluation adopted confined itself to using lecturers and students of databases and information systems in the Department of Computer Studies, University of Glamorgan. The majority of the lecturers used had a number of years of industrial experience in database design work. Also, a number of the students had previous industrial experience of data modelling.

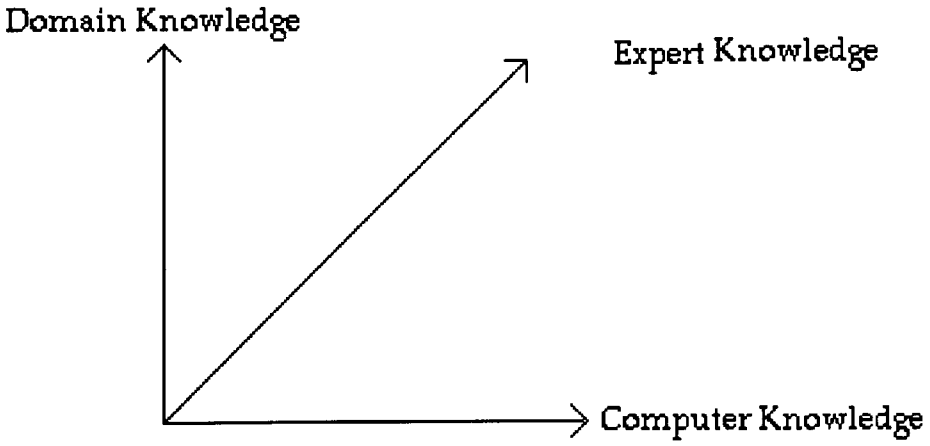

Figure 8.4: Domains of User Knowledge [Nielsen 1993].

The ideal training for our subjects would have been hands-on experience of the prototype to familiarise subjects with manipulating the user-interface as well as performing real-life integration tasks to arm them with the techniques of schema integration itself. Given the time constraints on the work, subjects were given informal training by way of a demonstration so that they could use the tool, as well as the concepts underlying the IBIS scheme and how they are used in SISIBIS for deliberation.

In total 16 subjects (excluding the author who assumed the role of facilitator) took part in the evaluation. Subjects were put into groups of two (small groups made it easier to observe and take notes about sessions) and given the same integration tasks. This gave us a total of 8 groups. Since we had 4 test conditions (same SISIBIS features and tasks), two sessions were conducted for each condition. The data used for analysis was taken from the average obtained from the two sessions. This was an attempt to smooth out random differences between groups.

An attempt was made to match subjects in terms of their prior exposure to schema modelling and where possible schema integration. This was done in order to:

a. reduce the effect of variation in abilities within a group.
b. ensure that participants deliberated freely and confidently without the fear of being dominated by more able co-participants.

However, such assignment of subjects to groups introduces biases towards certain groups due to the variation in abilities between groups. Random assignment would help reduce this bias, but it introduces the undesirable situation addressed in (b), thus possibly defeating the essence of collaboration. However, we believe that our selection of subjects from the areas of databases and information systems helped, to an extent, to control this variation. Although the author assumed the role of facilitator in each session he did not actively take part in the deliberations.

3.2. Evaluation Tasks.

[Nielsen 1993] states, as a basic rule for sampling evaluation tasks, that tasks should be as representative as possible of the uses to which the system will eventually be put in the field. Further the tasks needed to be small (but not too small to be trivial) enough to be completed within the time limits of the sessions. Our tasks are based on the classic example given in [Navathe et al 1986] whose scenario is a publishing company recording information about publications. We split this material up into five integration tasks increasing in size. The size of the tasks increased at each step either by requiring structural transformation of concepts (e.g. attribute -> entity) or increasing the number of concepts in the subschemas. We quantified the size of the integration tasks by assigning a cognitive weight to:

a. a concept name (attribute, entity or relationship) of 0.25 (names do not, but often give initial hint to, equivalence),
b. an entity of 0.25 (for name) + total weight of its attributes,
c. a relationship of 0.25 (for name) + 0.5 (for member entities) + total weight of its attributes.
d. a transformation of 1 (0.5 for identification of need and 0.5 for transforming).

Thus task size was a major (independent) variable in the integration tasks. The subschemas involved in these tasks were however simple and small so that the participants could easily comprehend them and complete the integration tasks within the allotted period of one hour. Since idea generation may be time-consuming, we acknowledge that the time period of one hour was a limiting factor. It was however impossible to get greater time commitment from the subjects (lecturers and students) due to academic commitments. Section 8 discusses an additional evaluation conducted over a period of two weeks.

While increasing the size of integration tasks was a necessary test condition, it also served the important role of giving the subjects confidence as the session progressed. The very first task consisted of only two entities and no relationships in order to guarantee early success and boost subject's morale. To ensure that subjects felt they had accomplished something, each successive task was an extension of the previous one. The tasks encompassed the five possible combinations of concept types during E-R based schema integration illustrated in Figure 5.6 (c.f. chapter 5), namely:

a. attribute - attribute.
b. attribute - entity.
c. entity - entity.
d. entity - relationship.
e. relationship - relationship.

The subjects were given a document (shown in Appendix 2) providing the evaluation guidelines of the session prior to the sessions. The document described a realistic hypothetical universe of discourse and its information areas. It also included E-R diagrams of the subschemas for the five integration tasks shown in Figures 10.1(a) - (e) of Appendix 2. In addition the document also described an example of a completed integration task together with the reasons for the integrations (Figure 10.2). A sample integrated schema

resulting from the integration was also given (Figure 10.3). It was hoped that this would give subjects time to generate some integration ideas before the sessions started.

4. Evaluation Sessions, Data Capture and Analysis, and Hypotheses.

Below we discuss the evaluation sessions carried out and the hypothesis that they were designed to test. We start with collaboration, followed by complexity.

4.1. The Evaluation of Collaboration.

The aim of this part of the evaluation was to test the thesis that SISIBIS as a problem-resolution system enhances collaborative schema integration.

i. The Evaluation Session.

The evaluation plan employed two groups using the same input schemas. One group (group A) conducted integration without a shared IBIS workspace (Figures 8.2(a) - (b)). The other group (group B) conducted integration with a shared IBIS workspace (Figures 8.2(c) - (d)). The subjects worked under the following conditions:

a. Group A had access to the facility supporting concept integration i.e. they only made assertions, resolved and merged them as a group.
b. In addition to the facilities provided to those in (a) above, Group B had access to the deliberation facility i.e. they could annotate the issue-base by way of explanations, annotations, data scenarios, questions and answers.

ii. Data Capture and Analysis for Collaboration.

The two groups were compared in terms of:

a. the times it took to complete a series of integration tasks. Data capture software was embedded into the tools to record the time at which significant events occurred. Consideration was given to the time it took Group B to type-in their annotations so that this time was eliminated from the comparison. The rest of the time was considered to be idea-generation time.
b. the production of a more satisfactory integrated schema. A more satisfactory integrated schema is one that has less redundancy (i.e. less duplication of concepts) and that is not 'lossy' [Desai 1990]. The concept of 'lossiness' is adapted from work on relational decomposition. A decomposition process

is defined as lossy if some attributes in a decomposed relation cannot be reconstructed from the relations resulting from the decomposition. Relational decomposition is a top-down process. In contrast, schema integration is a middle-out process. However, we use the concept of 'lossiness' in a comparable way for schema integration. Instead of thinking in terms of decomposition and attributes, we think in terms of integration and semantics. Redundancy was measured by analysing the integrated schema and counting duplicated concepts. Lossiness was measured by counting the number of lost semantics after integration i.e. semantics that appeared in the input schemas but not in the final integrated schema.

iii. The Hypothesis for Collaboration.

We expected:

a. Group A to complete their integration tasks faster than Group B for smaller schemas, and longer for larger schemas as illustrated in Figure 8.5. This expectation is due to the fact that Group A did not enjoy the co-authoring facilities enjoyed by Group B. Coauthoring introduces two tangential factors:

   i. users are not forced to focus on the same problem (e.g. assertion).
   ii. the overhead in gleaning other participant's points of view.

   Small schemas reduce the effect of factor (i) because of the corresponding reduction in the number of problems, effectively forcing users to focus on the same problems. So factor (ii) dominates, limiting group B's progress. However, for larger schemas the problem space increases. As a result the rate at which assertions are submitted increases and the overhead in aspect (ii) becomes beneficial in this respect. Thus the rate of progress swings towards group B.

b. Group B to produce more satisfactory schemas. This expectation is due to greater semantic consensus (than for group A) resulting from documented concerns.

4.2. The Evaluation of Complexity.

The aim of this part of the evaluation was to establish whether the link-node architecture of hypermedia facilitates the management of the complex inter-relationships among database design documents. The database design documents in this context are E-R dictionaries, integration dictionaries and

annotations arising from schema integration deliberation.

i. The Evaluation Session.

The evaluation plan employed two groups using the same input schemas. One group (group A) conducted integration without direct links to individual concept nodes in the subschemas and annotations (Figs 8.2 (b) - (d)). The other group (group B) conducted integration with direct links to individual concept nodes in the subschemas and annotations (Figs 8.2 (a) - (c)).

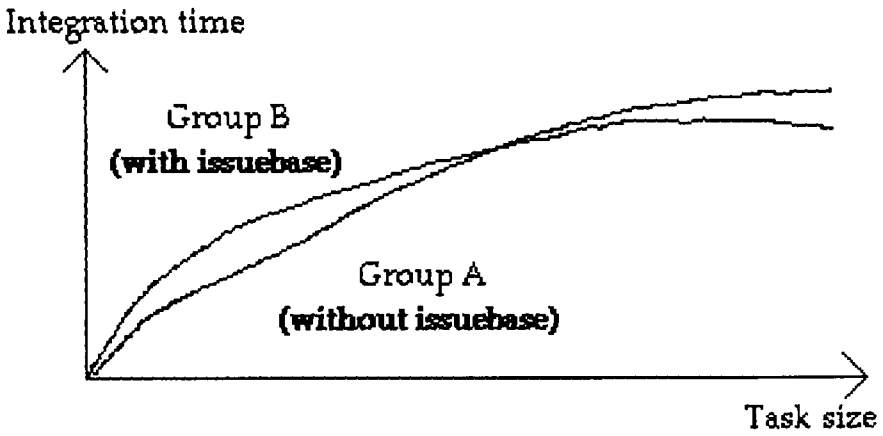

Figure 8.5: Expected Task Size vs. Integration Time Graphs : With and Without An Issue-base.

ii. Data Capture and Analysis for Complexity.

The data captured for these groups was both qualitative and quantitative. Qualitative data was captured through observation of the sessions mainly to collect any data management problems that subjects encountered. The management issues observed included the speed and flexibility of accessing concepts, and the creation and modification of annotations and the rationale of integration. Quantitatively, we compared and contrasted the two sets of data in terms of the number of attempted duplication of assertions. The data was captured by recording all traversals within and between the hyperdocuments used in the integration tasks.

iii. The Hypothesis for Complexity.

Qualitatively, we expected:

a. group A to have more management problems than group B. For large subschemas, the speed and flexibility of accessing concepts in subschemas will be inferior compared to B.

b. no difference in the creation and modification of annotations.
c. the flexibility of modifying the rationale of integration to be more pronounced for group B than for group A.

Quantitatively, we generally expected more attempted duplication of assertions for group A than for group B.

5 . Evaluation Results.

Tables 8.2 and 8.3 show the data collected for the sessions for the completion time of integration tasks, and collaboration and complexity respectively.

Table 8.2 shows the task completion times for each of the two participants in our subject groups. Both participants in the groups with no issue-base managed to complete all the five tasks in the allotted time, while those with an issue-base only managed to complete three. This is in agreement with our expectation for small schemas (since our schemas are relatively small). Consequently our comparisons are based on the first three tasks only.

Group (Condition)	Number of tasks completed by:		
	Participant 1	Participant 2	Both
A (No issue-base, with direct links to subschema nodes)	5	5	5
B (No issue-base, without direct links to subschema nodes)	5	5	5
C (Issue-base, with direct links to subschema and annotation nodes)	4	3	3
D (Issue-base, without direct links to subschema and annotation nodes)	3	3	3

Table 8.2: Task Completion Data for Integration Tasks.

Table 8.3 tabulates the results collected and analysed to test our hypotheses for collaboration and complexity. It shows (for the completed tasks) the completion time for each task (the larger of the two participant's times), the total number of assertions, annotations and different assertions sharing some concept(s). It also shows the semantic discrepancy between the subschemas and the resultant integrated schema.

5.1. Analysis for Collaboration.

We analysed these results by drawing and comparing the time vs. size graphs of the data for the groups with and without an annotated issue-base as discussed in section 2.1. We acknowledge that conducting statistical significance tests on the data might have helped to formalise our analysis. However, our subject sample was too small for such tests. We therefore analysed the behaviour of each individual graph for intra-group analysis, and compared the behaviour of the graphs relative to each other for inter-group analysis. We assumed that relative intra-group and inter-group characteristics did not change during the course of the sessions.

Figures 8.6 and 8.7 represent the results for the groups with, and without, an annotated issue-base respectively. The session conditions for Figure 8.6 had direct links to nodes in the subschema dictionaries, while the links were not direct for Figure 8.7.

Group	Completion Time for Task(Secs):					Count for Tasks 1-3 of		Count for Tasks 1-3 of		
	1	2	3	4	5	Assertion	Annotation	Altnvs	Dupls	Dscrpnces
A	553	1123	1335	1769	2640	54	-	-	-	0
B	955	1540	1867	2434	3080	44	-	-	-	+2
C	1002	1793	2362	-	-	22	5	4	4	0
D	1148	2175	2820	-	-	22	4	3	2	-1

Key: Altnvs=Alternatives, Dupls=DuplicationAttempts, Dscrpnces=SemanticDiscrepancies

Table 8.3: Collaboration and Complexity Data for Integration Tasks and Groups.

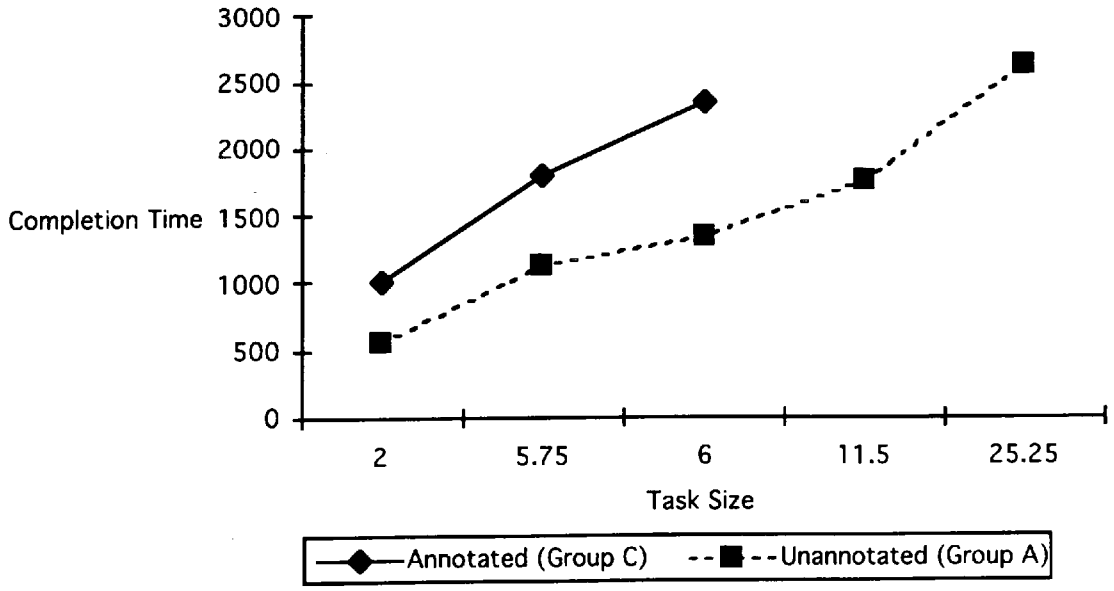

Figure 8.6: Completion Time vs. Task Size for Groups With Direct Links.

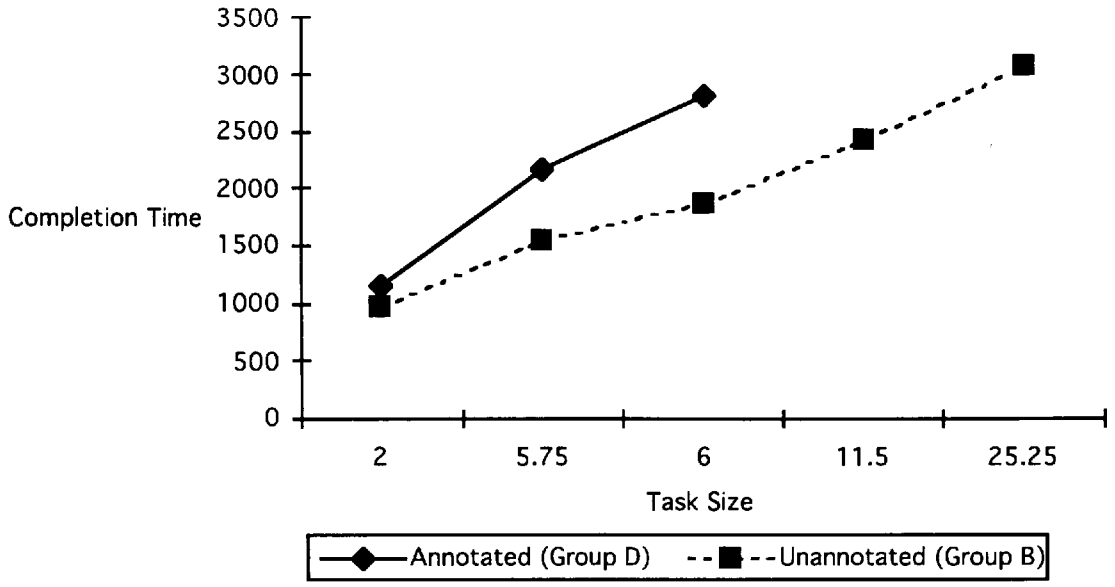

Figure 8.7: Completion Time vs. Task Size for Groups With No Direct Links.

Figures 8.6 and 8.7 demonstrate some consistency with the hypothesis discussed in section 4.1 for the first three tasks. However, we cannot compare the results for the two groups beyond the third task. After Task 3 the groups without an annotated issue-base take more than proportional time (to task size) to complete their tasks. longer as the task size increases. We attribute this discrepancy with Figure 8.5 (for hypothesis) to matters of gleaning, analysing and managing data semantics taking prominence, so still offsetting any benefits of the tool.

Both graphs show that the groups with an annotated issue-base took a longer time to complete each task. We would expect so due to the time spent analysing other participant's assertions. Figure 8.8 compares the degree of use of the various aspect of SISIBIS for all the sessions. It is not surprising that the total number of assertions submitted by the groups without an annotated issue-base is comparatively less than that for the groups with an annotated issue-base. The reason is that there was no possibility for other members of the group to document their views (even if they had any). That is, there was no mechanism to capture and document such discussion, typical of pen-and-paper based integration. As a result, the rationale for some integration decisions was lost which may be dangerous for large-scale database design work in view of database maintenance and changes in personnel.

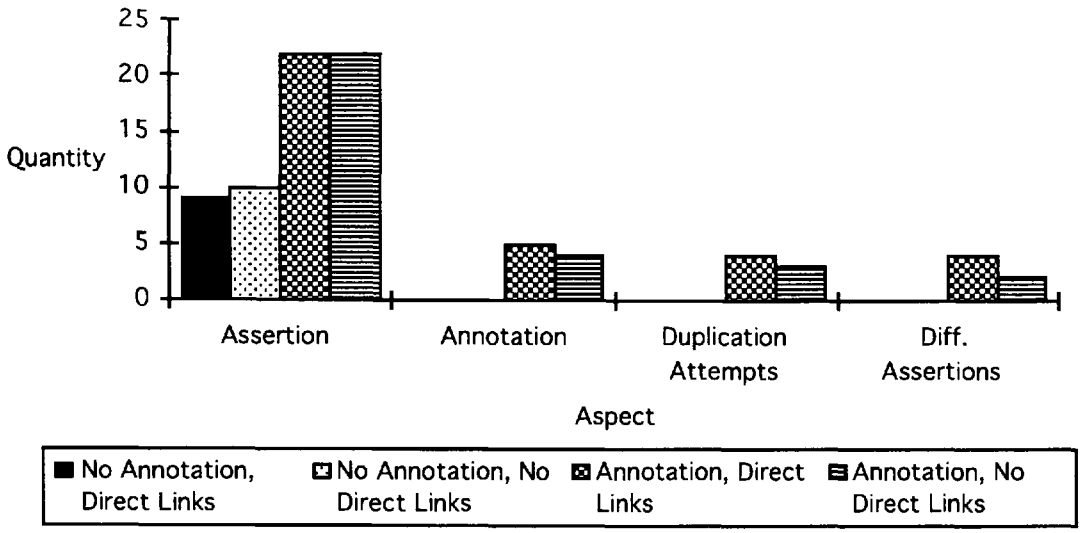

Figure 8.8: Aspect Usage Chart for the Different Groups.

The significant observation from Figure 8.8 is that the groups with an annotated issue-base annotated about a quarter of their assertions. We believe that since some assertions were not annotated, annotations were provided for those assertions that were not trivial. However, no annotations were provided following a request though we anticipated some annotations to emerge this way, especially for more complex schemas. The majority of annotations were either explanations, questions and answers. This suggests that comments and data-scenarios may not be frequently or directly used as individual nodes, but may be implied or embedded in explanations, questions and answers. Further, most of the annotations were linked to the 'why' aspect of proposals, suggesting that the rationale for equivalence (not how the concepts are to be merged) was regarded more highly. Observation of the annotations revealed that most annotations for proposals made reference to the attributes of the entities/relationships. Effectively, annotating issues (attribute assertions) may repeat what has been stated in the proposal's annotation.

The semantic discrepancy for groups A and C in Table 8.3 is 0 i.e. no subschema semantics were lost or and no redundancies were introduced. On the contrary, Group B introduced 2 redundancies, while Group D lost one subschema semantic. It is unrealistic to infer any conclusions from this set of data as the data is to a large extent dependent on the thoroughness of the groups as well as time constraints. Nevertheless we believe that groups A and C did appear to benefit from the presence of direct links in producing better integrated schemas.

5.2. Analysis for Complexity.

Our analysis for complexity falls into two categories: quantitative and qualitative.

5.2.1. Quantitative Analysis.

As for collaboration, we analysed the results by drawing and comparing the time vs. size graphs (Figures 8.9 and 8.10) of the data for the groups with and without direct links, and where appropriate annotation nodes.

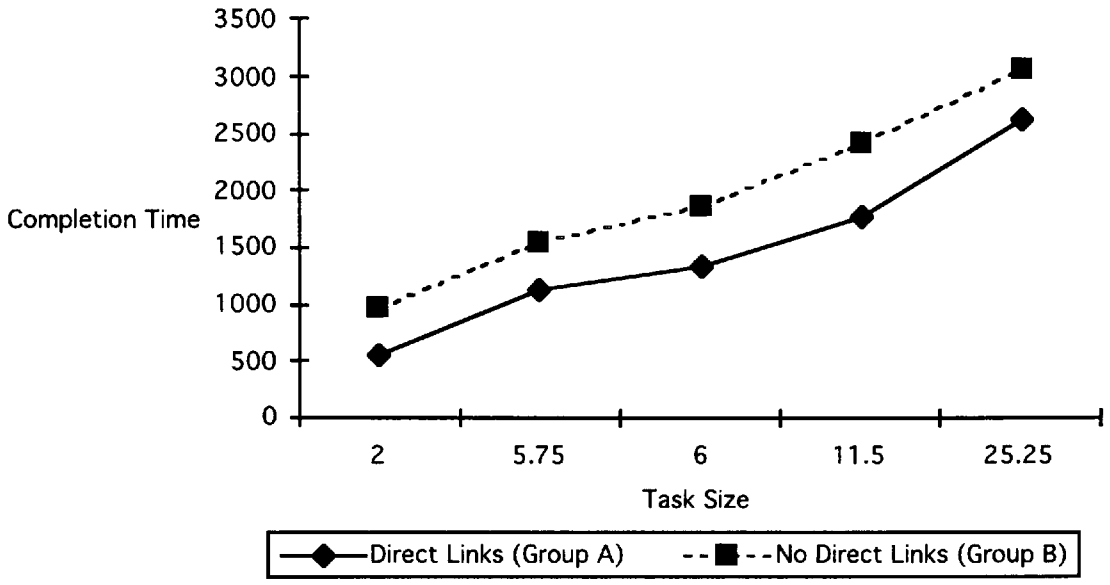

Figure 8.9: Completion Time vs. Task Size for Groups With No Annotation.

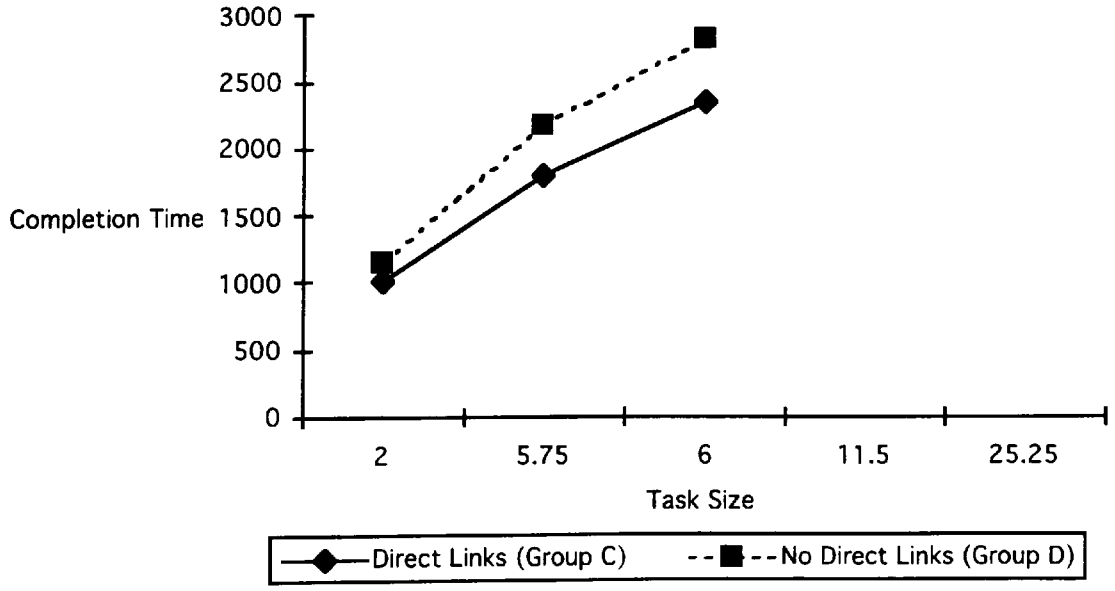

Figure 8.10: Completion Time vs. Task Size for Groups With Annotation.

Figures 8.9 and 8.10 demonstrate some consistency with the hypothesis that the presence of links positively supports schema integration. The gap between the two graphs (i.e. the time difference) progressively increases in Figure 8.10 (in comparison to Figure 8.9). We attribute this to the corresponding increase in the link space and the cognitive requirements resulting from the presence of annotations.

5.2.2. Qualitative Analysis.

One of the groups without a shared issue-base used pen-and-paper to connect up attributes that were suggested as equivalent. It appeared this became necessary to allow views from both participants to be recorded as soon as they were suggested. This seemed to serve two purposes:

a. identifying similarities and differences in their individual assertions.
b. outlining assertions before actually submitting them (especially for entities and relationships).
c. (quick) identification and discussion of the implications of actions. Occasionally, some subjects reservedly agreed to certain assertions.

On the contrary, subjects with a shared issue-base presented their assertions to the issue-base directly for deliberation. This, according to some subjects:

a. gave them time to organise their thoughts and work.
b. allowed them access to other people's ideas and have their ideas validated by others.

Also, they submitted proposals to which issues (attribute assertions) were later attached. We believe that the node-link mechanisms of hypermedia supported incompleteness (details for proposals- attribute assertions- could be defined later and incorporated into the argument). Further, the fact that some issues were defined by a subject different from the submitter of the proposal points to the utility of hypermedia in realising co-authoring. We can take the same view for explained agreements to given assertions. Thus the shared issue-base served the same purpose as the use of pen-and-paper for the other group, only that assertions took longer to resolve and the rationale for an assertion was documented.

All subjects found accessing the dictionary entries for concepts extremely helpful. They noted that it helped them identify equivalences and augmented

their thoughts during verifying or validating tentative assertions. However, we attribute the helpfulness to organising data according to nodes and links: the availability of a link to a node which effectively answered the majority of questions the participants needed to know about a concept.

6. Informal Evaluation.

In discussing the evaluation of a related system, an Intelligent Learning Environment, [Twidale 1993] argues for early informal evaluations. He cites the disadvantages of a formal (controlled) evaluation to include:

a. *'Rigorous experiments are large, slow and costly'.*
b. *'A controlled experiment only really measures one thing'.*
c. *'A controlled experiment produces averaged out figures of overall performance'.*
d. *'Unexpected interactions may lead to misleading results'.*
e. *'The effect of the interface'.*

In view of the limitations imposed on our formal evaluation, we carried out two such (informal) evaluations:

i. iteratively during development of the system.
ii. to supplement the formal evaluation discussed, we worked collaboratively in integrating individually developed schemas of the same domain over a longer period of 2 weeks.

i. Iterative Informal Evaluation.

The iterative informal evaluation was in the form of demonstrations conducted by the author. The major comments provided by commentators during the demonstrations are:

a. The CSCW nature of SISIBIS: While SISIBIS's organising framework for deliberation was useful, the tool covered only a subset of CSCW concepts. Additional support for commitment (of attendance or response to a raised problem), documentation of informal meetings and human communicative gestures may be helpful. While useful in documenting meetings, such an approach was felt to lead to an expansive network of nodes lacking in concision, cohesion and possibly documenting unnecessary digression.

b. The Need for Design Maintenance: It would be desirable if modifications to earlier concept integrations were propagated up the binary-integration hierarchy (depicted by arrows (a) in Figure 8.11). The ability to automatically reference or link assertions and annotations defined at different stages of the binary integration strategy would also be desirable (depicted by arrows (b) in Figure 8.11).

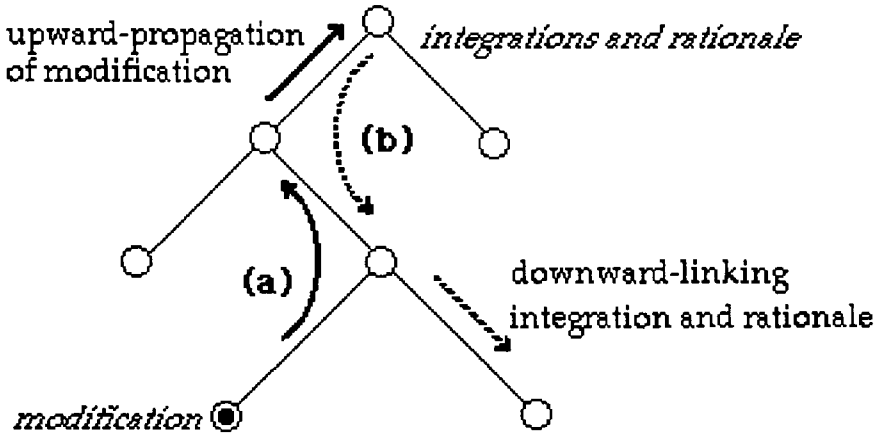

Figure 8.11: Propagation of Modifications , and Linking of Integrations and Rationale.

c. Consideration of Operations On Data: The tool's utility may be compromised by shortcomings inherited from the (extended) E-R model e.g. lack of modelling operations (processes) on data. Looking at data from the operations point of view might reveal some collision of operations on the same data. We may therefore decide not to merge the concepts in question or be made aware of situations that may compromise the integrity of the resultant database if the concepts were merged. [Larson et al 1989] attempted to address this issue by considering the temporal nature of equivalences. We, however, feel that this issue is better addressed in process models instead of data models (and the findings propagated to the data model) because:

i. data is comparatively more stable (structurally and semantically) than the operations performed on it and as a result the models resulting from integrations are more likely to be unaffected by the dynamicity of operations on data.
ii. operations have semantic problems of their own which can constitute a challenging research project [Winter 1994]. For instance, [Larson et al 1989] introduce the idea of strong and weak equivalences, and distinguish

between equivalences occurring for all time from those occurring at a given time. A strong equivalence implies that both retrieval and update are possible against the integrated schema, while only retrieval is permitted for a weak equivalence.

d. The Effect of Abstraction Hierarchies: Even though the abstraction mechanisms of aggregation and generalisation were understood in principle, problems arose in determining the optimal level at which an integration is done in an abstraction hierarchy. We believe this problem is attributable to focusing on one end of the hierarchy instead of a hierarchy-wide search. This obviously introduces a cognitive overhead in requiring the integrators to memorise the structure and semantics propagated down the hierarchy. Analysing for equivalences in a bottom-up manner (i.e. based on attributes: provided inherited attributes only appear in the generalising entities) may reduce this cognitive overhead. However, it is likely to be counter-productive by hiding the detail necessary to ascertain equivalence.

ii. Supplementary Informal Evaluation.

This part of the evaluation was conducted with a view to compare and contrast it with the shorter formal evaluations we have described. [Nielsen 1993] describes an approach to evaluation called heuristic evaluation. A heuristic evaluation may be conducted by expert users via applying their knowledge of established principles and techniques to find problems without the need to involve experimental subjects. In view of the limitations imposed on this work, this part of the evaluation can be viewed as a 'heuristic' evaluation.

For SISIBIS the evaluation was conducted by the author and a member of the supervision team of the project. For HSMS feedback was sought from a lecturer with E-R modelling experience. A common information area was chosen, course administration in a university, and individual schemas of it were produced. We then conducted a full integration of the two schemas using SISIBIS. The evaluation highlighted the following:

a. The lack of abstraction mechanisms for relationships: integrating specific entities with general or aggregate entities presented semantic integrity problems. The problems stemmed from use of aggregation and generalisation as modelling constructs for entities only. For instance, integrating a less abstract entity and a more abstract entity (such as an entity

at the top of abstraction hierarchy) effectively handed-over its specific relationships to that entity and its generalised entity. However, such relationships did not exist in the information area. A related situation is illustrated in Figure 8.12.

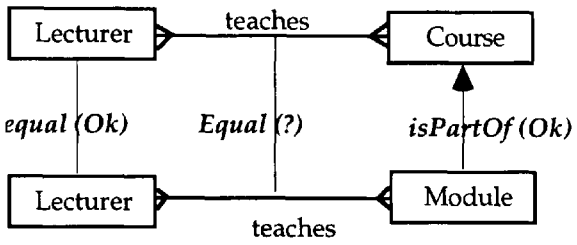

Figure 8.12: Abstractions and Relationships Integration Example.

The following assertions were made: 'Lecturer equal Lecturer' and 'Module isPartOf Course'. These assertions were then passed-on to the relationships 'Lecturer teaches Course' and 'Lecturer teaches Module' tempting the assertion that these two relationships were equal. Closer examination revealed that the semantics captured only tells us which lecturers teach which courses. However, we lose the specific (and more important) semantic information which tells us which module (of a course) is taught by which lecturer. In view of this, we are convinced that the need for collaboration is heightened by the use of aggregation and generalisation. In the light of this perhaps the demonstrator should be given the opportunity to automatically make assertions (after humans have submitted all theirs) for concepts with the same name.

b. Lack of a direct way to specify conflicts: our approach of asserting equivalences between concepts directly addresses the issue of synonyms. We used the philosophy that if two concepts have not been resolved as equivalent and their names are the same, then they were homonyms. The tool automatically tags one of the entity names with a character (@) to make them different, thus allowing assertions for them to be entered at the next iteration. This had the negative effect of increasing the number of iterations. A direct way to assert such conflicts is therefore desirable to reduce the number of iterations. Alternatively, the tool searches for similar concept names and proposes them as equivalent, triggering deliberation.
c. Feedback on HSMS: Feedback on HSMS, the conceptual modelling component of the demonstrator, was sought from a lecturer with a large

amount of E-R modelling experience. The lecturer also has wide experience in the use of a PC-based CASE tool: Select SSADM [Select Software Tools 1993]. His task involved modifying the integrated schema's dictionary and produce diagrams with the diagrammers.

A feature found lacking in HSMS was the specification of foreign keys. However foreign keys were beyond the immediate scope of the project as entities would undesirably exhibit some equivalences during schema integration.

The flexibility in switching between the dictionary and the diagrammers was positive, unlike in Select where the diagrammer is the seat of all interactions resulting in the lack of a direct way to browse the dictionary. However, the ability to switch between the dictionary and the diagrammers highlighted the need for universal editing operators. Data analysts might prefer the flexibility to effect modifications in both the dictionaries and the diagrammers, and have them reflected in both. At present the diagrammer only allows the modification of relationship constraints, but dictionary modifications are propagated to the diagrams. We however recommend that the best place to decide on (and effect) a modification is in the dictionary because the decision is more likely to be better informed there due to the available detail, as opposed to mere diagram symbols.

The clustering mechanisms of HSMS's E-R diagrammer were viewed as an undesirable overhead, often disorienting and slowing down the diagramming process. This may have been due to the concept of logical horizons being too theoretical. On diagramming, the automated checking of relationship arcs intercepting entity symbols was positive, a feature totally lacking in Select. Any such intersections are allowed to occur in Select but are removable (interactively by the analyst) via the creation of so-called 'waypoints' (points where arc segments meet). It was felt that HSMS and Select address the problem of arc-labelling in different ways. While HSMS positions relationship labels on arc segments, Select allows positioning the label on either side of the arc. However, while the side could be easily changed, it was impossible to conveniently place the label anywhere along the relationship arc in Select, unlike in HSMS. Understandably this restriction is due to the use of relationship-roles (i.e. a relationship arc is labelled on either-end to depict the role of the corresponding entity in the relationship in question). HSMS avoids relationship-roles by attaching a directional symbol (an arrow) to the relationship symbol. It was also commended that the tracing of a relationship was visually intuitive as the

path of the relationship arc(s) is made apparent as drawing takes place. After the E-R diagrams were drawn, the facility for highlighting clusters was noted as handy because the major entities of a universe of discourse (or in the diagram) could be identified. While the E-R diagrammer was appreciated and commended, there was cautious criticism expressed as to the need for the F-D diagrammer. It was viewed as more useful during the normalisation process of the next phase: logical modelling.

7. Some Usability Issues Arising from the Evaluation.

This section itemises some of the major usability issues that arose during the evaluation sessions, namely:

a. Schema Integration Terminology: some subjects occasionally had problems coming to terms with the terminology for equivalence types (e.g. Equal, Overlaps) and merge methods (e.g. Union, Meet).

b. The User-Interface: A number of comments were provided about the user interface.

One subject felt the interface was too cluttered with icons and graphics, and that the windows were too small. The small window size was inherited from the initial development platform of the tool: a Macintosh SE/30 with a relatively smaller screen. The cards had to be correspondingly smaller. When the development shifted to a larger-screened machine, a fundamental decision was taken to retain the current card size for portability of the prototype to the Macintosh SE/30 which apparently has the smallest screen size. Further, large card sizes tended to bring up memory problems on the machine.

Yet other subjects felt an E-R diagram-based schema integration interface would further enhance the usability of the demonstrator. However, this approach seems desirable, faster and practical where the diagram is not split into nodes or the diagram is hierarchically structured because equivalences will be determined in the context of the whole picture. The overriding factor for not adopting this approach was that diagrams inherently provide much less data semantics than the dictionaries. We acknowledge, however, that diagrams offer a broader basis for suggesting equivalences than the list of concepts that appear in pop-up menus.

c. Learnability: Opinion of ease of use was varied, but the majority of subjects felt the system was not very intuitive. Most of those that felt the demonstrator was intuitive were familiar with WIMP (windows, icons,

mouse and pointers) interfaces and felt the demonstrator had a good human-computer interface. All subjects felt on-line help would greatly enhance the usability of the demonstrator. Suggestions were also made to provide an advice facility e.g. suggesting the 'next task', indicating that some subjects found the interface to be 'hyperfunctional' (too much functionality on one window), yet the functionality was not readily available. This is a direct consequence of an attempt to reduce user-interface clutter (by delegating functionality to pop-up menus appearing after pressing the mouse over an icon).

d. Safety and Confirmation Features: The majority of the subjects were critical of the inability of the demonstrator to cope with multiple assertions involving the same concept and provided by the same participant. Also, most subjects were critical of the lack of confirmation and safety/warning features in the demonstrator. In particular, they felt the most common error message (of duplicated assertions or multiple assertions involving the same concept and submitted by the same participant) was too cumbersome. This message definitely needs to be made more acceptable.

e. Compatibility Checks for Issues: one subject felt more stringent compatibility checks were needed for the concepts involved in an integration e.g. the attributes 'sex' and 'name' of a student could be merged. The observation is valid. However, the demonstrator validates such assertions based on the underlying type of both attributes. Because the notion of data types (integer, character, real etc.) is very broad, the demonstrator only performs base-type validation and it is up to the integrators to go further with the validation. Ideally, we expected the participants to deliberate over such an issue using the system (which is the purpose of the demonstrator anyway).

f. Message Control: Some mechanism to record received messages for later attention is desirable. For instance, deferring response to a question is necessary to avoid distraction. To achieve this conveniently, one must remember the aspect ('why' or 'how') of the assertion to which the annotation was attached. In addition, some annotations may not be attached to these aspects, hence the need to address annotations directly without going through their originating assertion becomes paramount. It is therefore desirable to allow the recipient of messages to instruct the tool to save the message so that they can address it at a time of their own choice. This is more so for collaborative schema integration as idea generation may take a long time while other information is being gleaned. Such a facility demands a context-sensitive index of proposals, issues or annotations to

help participants quickly find the artefact they want to address.

g. Performance: The speed of execution of SISIBIS was very slow, a situation further compounded by the presence of a lot of traffic on the network. This is however a limitation of the underlying hypermedia platform, HyperCard, which is intrinsically slow. Also, the local-copy option adopted in SISIBIS slows down the system each time local copies of files are made. Attempts were made to improve the performance of the demonstrator by developing computationally intensive modules in Pascal and by making copies of only those files that were modified since the last copy.

h. Length of Concept Name: Some subjects were correctly critical of the limitation of the length of entity and relationship names to 10 characters (a limitation imposed to enable the dictionary nodes to be properly tagged with the concept nodes in order to facilitate linking and searching).

i. Subschema vs. Asserted Semantics: Two unexpected uses of SISIBIS were observed during the evaluation.

Firstly, arguments were poorly constructed, and most annotations were not enriching the assertion or argumentation. Our expectation was to find explanations laid out along the following lines: *'The equivalence between* X *and* Y *is because of* a, b, c,....' for the 'why', and *'If we merge them as* Z *we can achieve* d *in our schema, or we can reflect aspect* e *of the domain'* for the 'how'. Instead, annotations such as *'Yes,* X *and* Y *are the same'* and *'I agree with you'* were common but did little to substantiate the argument.

Secondly, one group without an issue-base at their disposal occasionally made verbal references to previous assertions they had submitted. Statements such as *'if we said A is equal to B because ..., then ..."* were often used. This is indicative of the importance of recording arguments. In this respect, we believe that had the issue-base been at their disposal it would augment their memory positively. However, such references were also noticed while searching for equivalences. This was correct usage of SISIBIS insofar as avoiding circular assertions and progressing with the integration task were concerned. However, the subjects risked circular deliberations by mixing assertive semantics (constructed during deliberation) with established semantics (in the input schemas). Such use of the demonstrator was unexpected and is negative. We believe, however, that in a realistic setting subjects will come to understand that their assertions were correct only when selected as the resolution by the facilitator.

j. Collaboration and the Iterative-Binary Strategy: It was also observed that the pragmatics of realising the iterative-binary integration strategy would

present delays and increase the number of iterations to complete an integration task. This means that other members of the group had to wait for the facilitator to define resolutions, rationales and implement mergers before enacting the next iteration. Further, participants needed to be kept informed of the level of integration. In summary, the demonstrator lacked the facility to support co-ordination between members of each group. Further, we postulate that the more time spent waiting for other participants' contributions, comparing and contrasting them may reduce the acceptability or utility of the demonstrator.

8. Problems with the Evaluation.

A number of problems were experienced with the evaluation. These include getting adequate time commitment from the subjects, limited and variable schema integration literacy on the part of the subjects and getting subjects to adapt to organising ideas according to the IBIS scheme. Such problems are however not unique to the current project [Yakemovic and Conklin 1990, Grudin 1994].

Firstly, the nature of collaborative work requires considerable time commitment from the subjects. Ideally, an evaluation of a problem-resolution tool would be done over a long period so as to give the subjects time to generate integration ideas, deliberate over them and for the facilitator to come to a comprehensive assessment of the deliberations. For instance, [Yakemovic and Conklin 1990] describe an evaluation of the use of an Issue-Based Information System (IBIS) during a development project. The observations were made over a long period (24 months), thus enabling them to generate a large quantity of issues, focus on a number of collaborative work aspects (social acceptability, problems and benefits of accessing IBIS data) and how introduction of the tool influenced the development process itself. Such a summative evaluation was clearly unattainable in the academic setting available to us.

Secondly, the subjects need some time to get experienced with the tool, especially the idea of organising information into self-contained units. This might be tantamount to getting accustomed to a different working practice. Further, the techniques of classifying and organising deliberation aspects around the IBIS framework needs to be built into the working practices of would-be users. [Yakemovic and Conklin 1990] cite getting users to know '... *how to "reduce" normal conversations to issues, positions and arguments*' as a major problem. They add that some subjects were overwhelmed by the

nomenclature of their IBIS-based tool and refused to take part. We had similar problems with a subject who, after turning up for a session, refused to take part because he needed prior coaching in order to accomplish the task.

Thirdly, schema integration literacy proved another source of difficulty. Schema integration is currently a paradigm largely confined to research laboratories. It is not surprising therefore that despite its wide practical implications not many practitioners or academic institutions emphasise it in their development practices or curriculum. Hence there were problems in getting subjects who would quickly produce the results i.e. who knew the problem at hand and knew how to identify discrepancies.

More recently, [Grudin 1994] aptly summarised these issues by citing the difficulty of evaluation as one of the eight specific problems areas of collaborative environments research. He notes that:

a. evaluations of multi-user environments are more difficult than for single-user applications because of the differences in roles, backgrounds and preferences of a number of users.
b. the evaluation takes longer as group interactions unfold over a long period.
c. because of the large number of factors that affect collaborative work (social, economic, motivational, political), it may be risky to generalise from experiments. He adds *'establishing success or failure is easier than identifying factors that brought it about'*.

In view of these factors, the evaluation can be viewed as necessarily preliminary. The next section discusses what we consider to be an ideal evaluation of the work.

9. Description of an Ideal Evaluation.

In the summary section to this chapter, we described the necessarily preliminary nature of our evaluation. In this section we discuss some of the features of an evaluation we would wish to conduct given more time and resources.

An ideal evaluation would be one:

a. which is conducted in a natural setting i.e. subjects are professional data analysts and subschemas are large. As a result, the evaluation period must be considerable.
b. whose subschemas (task sample) are not prescribed, but are developed by

members of the integration group themselves (subject sample).

c. which considers all components of our I-CAISE (Integrated-CAISE) demonstrator: HSMS, the diagrammers and SISIBIS (Figure 8.13).

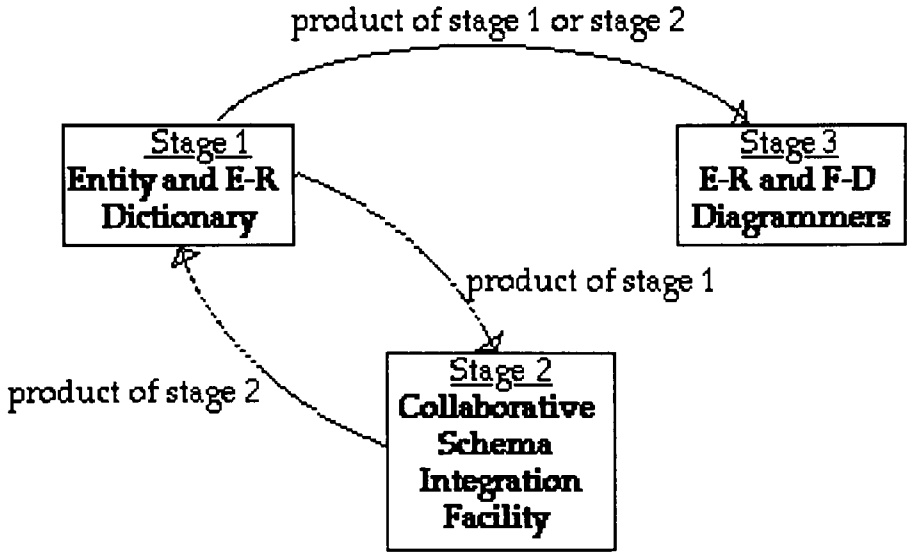

Figure 8.13: Interdependencies between Demonstrator's Components for an Ideal Evaluation.

d. considers trade-offs between the task completion time and the quality of the resultant integrated schema. For instance, [Moody and Shanks 1994] provide a framework for evaluating the quality of data models in terms of the quality attributes and the agents for measuring them. The quality of the resultant integrated schema may be measured in terms of: completeness, simplicity, flexibility, understandability and integration with the corporate data model. The ideal agents for assessing these quality attributes include end-users, senior management, industry experts, data analysts and data administrators (DBAs). Table 8.4 shows a matrix indicating which agent(s) may be responsible for which quality attribute. Thus a larger and wider sample of subjects is required. The assessment of schema quality in our evaluation only focused on completeness.

10. Conclusion.

The chapter has described an evaluation of the demonstrators discussed in this thesis. While the results obtained for SISIBIS, our collaborative schema integration tool, are generally consistent with our hypotheses they are only tentative. The samples of integration tasks and evaluation subjects are cases in point. The size of the subschemas was probably too small to realise the benefits of SISIBIS as anticipated. Also, the level of deliberation is to a large extent

dependent on the number of participants taking part in the integration process. For instance, we expect the level of deliberation to increase with the number of participants.

	End-Users	Senior Managers	Industry Experts	Data Analysts	DBAs
Completeness	•	•	•	•	
Simplicity				•	
Flexibility		•	•	•	
Understandability	•	•		•	•
Integration					•

Table 8.4: Matrix of Schema Quality Attribute/Suitable Subject for An Ideal Evaluation - adapted from [Moody and Shanks 1994].

The limited familiarity of a number of our subjects with the tool and the technicalities of schema integration further questions the reliability of the results obtained from the evaluation. For instance, the concept of practice-factor may have contributed substantially to the results. That is, since our subjects did not get enough training in the use of the tool, the rate at which they executed later tasks may have been affected by the expertise (in using SISIBIS) acquired during earlier sessions.

The findings however indicate that collaboration seems to contribute to good schema integration work. It is noteworthy that 15 of the 16 subjects agreed to take part in further evaluation sessions. We believe this is indicative of the utility they perceive in SISIBIS. In addition, most subjects felt argumentation and its recording were the most useful features of the demonstrator.

The results for complexity indicate that hypermedia, as implemented in the HyperCard environment, appears to provide excellent opportunities for collaborative work by flexibly supporting incompleteness. However, better sharing facilities (for collaborative work in general) are needed to improve performance. Basic file sharing is inadequate as the sharing parameters defined for the file are applied to the nodes within the file. For collaborative work participants need 'local work spaces' defined by the boundary of the node in which they are. Collaboration therefore requires node-level (local) protection offering both read- and write-privileges to the owner, but only read-privileges to other participants. The very nature of hypermedia makes such 'node-level'

sharing protection conceivable.

The next chapter summarises the project and provides directions for further research, addressing some of the problems highlighted in this chapter.

Project Summary, Further Work and Conclusion.

Summary.

Hypermedia, a form of information representation and management based on the principle of connecting a number of nodes of self-contained chunks of information with links, has been widely used for the management of semi-structured data. We have investigated applying the distinct properties of this technology, as represented by HyperCard, to database design. This chapter summarises the investigation. The focus of the work has been on conceptual modelling, a database design phase comprising two stages: schema modelling (cf. Chapter 2) and schema integration (cf. Chapter 5). Our investigation has been done via the development of a hypermedia working prototype that supports both schema modelling and collaborative schema integration (cf. Chapters 6 and 7). Section 1 restates the basis of the project. Section 2 summarises the work done on the schema modelling prototype and the conclusions emerging from it. Section 3 summarises the work done on the collaborative schema integration prototype. It also discusses the results emerging from the work. Section 4 proposes future directions of research for this work. Finally, section 5 concludes the chapter and indeed the thesis.

1. Introduction.

This research has attempted to merge conceptual modelling activities and hypermedia facilities by developing an active CAISE (Computer-Aided Information Systems Engineering) tool for schema modelling, collaborative schema integration and documentation. The demonstrator currently runs on Macintosh hardware using Apple's HyperCard 2.0 and a suite of processing modules developed in Pascal.

The focus of the work has been on the design of databases for standard information systems. The thesis of the work is that:

a. Database design can generally be cast as an irregularly structured process.
b. Schema modelling produces design artifacts that are irregularly linked.
c. The logical relationships between schema components (e.g. entities) often necessitate non-linear reading of design documents.
d. Hypermedia is a technology suited to the representation and management of irregularity. In particular, its node and linking mechanisms suggest that it may be suitable for:

i. supporting the irregular nature of database design tasks.
ii. managing the complexity of database design documents.
iii. enhancing the collaborative nature of database design work.

The major distinguishing features of the author's work are that:

a. it supports and provides a framework for collaboration and deliberation, as well as documenting the deliberations.
b. integrations are based on semantic consensus, thus it is fundamentally not a knowledge-based system. It therefore offers limited automation, a feature which positively enhances collaboration.

2. An Assessment of the Applicability of Hypermedia to Schema Modelling.

The schema modelling prototype, HSMS, allows a data analyst to develop a schema based on the extended entity-relationship model (EERM) [Teorey et al 1986]. The main facilities of HSMS are an entity dictionary, a relationship dictionary, an E-R diagrammer, an F-D diagrammer, a query facility and a critiquing facility. A browsing facility is provided in each of the stated facilities.

HSMS benefits from hypermedia with respect to the:

a. Representation of schema components: Conceptual modelling work exhibits two main forms of data: text and graphics. These data forms are readily supported and represented as self-contained components representing either a single concept or an aggregation of related concepts. This data is semi-structured e.g. the number of attributes varies from concept to concept. Support for this semi-structuredness is vital in realising self-contained nodes representing an entity/relationship. Consequently, adding a new attribute need not necessitate modification of the screen layout.
b. Presentation of schema components: Our hypermedia environment (HyperCard) appears to offer good representation facilities for presenting schema components. Components are organised hierarchically to depict relative importance, lists and traversal paths. The idea of encapsulating text and graphics aids the representation of a notation. The interface of HSMS has however been accused of being too cluttered with information.
c. Flexibility during the process of schema modelling: Because hypermedia nodes are self-contained semi-structured chunks of information, schema components can initially be incompletely specified and then iteratively developed.

However, the given hypermedia authoring environment (HyperCard) proved inadequate for supporting E-R and F-D diagramming. While suggestions have been given as to the size of a comprehensible diagram [Feldman and Miller 1986], ideally data analysts would prefer unlimited drawing space. Unfortunately, the overall size of a node (i.e. the drawing space) is limited. Under the given restrictions, an attempt was made to segment diagrams by utilising an entity clustering facility. Clustering only reduced the size of the problem but did not solve it, and further it hampered the process of diagram production by demanding additional interaction from the user and additional management of the diagram elements.

3. Applying Hypermedia to Collaborative Schema Integration.

Schema integration is necessarily a negotiation process between the data analysts responsible for the production of the subschemas. An essential aspect of the negotiation process is to reach consensus i.e. establish the rationale for integration and articulate the schemas in readiness for integration. A hypermedia-based Issue-Based Information System (IBIS) [Rittel and Weber 1970] provides a framework within which such negotiation can be supported and the generated information captured and documented.

We developed SISIBIS to demonstrate the salient features of an environment for such negotiation. SISIBIS supports the technical and collaborative aspects of schema integration. The collaborative schema integration facility supports binary integration. A participant in the integration process makes a clear formal proposal for an integration between two selected concepts and gives an explanation. Other participants respond to the proposal, annotating it further with explained agreements/disagreements, questions, comments and data scenarios. This deliberation process attempts to capture the rationale behind the final integration decision.

We assessed the usefulness of SISIBIS via an evaluation which was necessarily preliminary in nature due to the various limitations imposed upon it. The initial results appear to suggest the potential of SISIBIS to positively support the work of DBAs and data analysts. Foremost, capturing design information is likely to improve the maintenance of the resultant database: *'Traditional maintenance has proven difficult in the absence of design information, maintainers often spend considerable energy trying to recover this design information before making changes. Capture and reuse of certain kinds of design information should consequently improve the maintenance process'* [Baxter 1992]. We might speculate that not only may

SISIBIS be used to prototype and 'evaluate' new organisational schemas, but it might be used to capture the rationale and essential elements of existing schemas for new information systems development staff. Consequently, the new recruits will need less formal training to master the organisational data space thus offering resource savings (manpower, time etc.). Similarly, new recruits may bring in new expertise to the development teams. Either way, SISIBIS acts as a mutual learning environment.

The nature of the binary approach to schema integration and collaborative work put conflicting demands on SISIBIS. It is desirable that participants see the result of an integration within a reasonable period of time after deliberating over it. On the contrary, solution-seeking collaboration requires that all possible solutions and their arguments are entered before the problem is resolved. However, this is usually not possible as ideas unfold over time, so often a problem is 'resolved', possibly triggering new ideas. While increasing the number of participants may help to increase the level of deliberation, the waiting time is also likely to increase. Perhaps this means that the adapted IBIS framework needs streamlining to reduce the overhead in supporting certain annotations (e.g. comments, data scenarios). Further, better facilities for sharing work (nodes) are needed as the conventional file sharing methods proved inadequate.

Our initial results suggest that hypermedia does appear suitable for the presentation and management of conceptual modelling documents i.e. it does appear effective in managing the schema dictionaries and the issue-base arising from the collaborative schema integration process. It was also effective in presenting data in these facilities, but its HyperCard implementation proved inadequate in representing diagrams. The idea of identifying clusters in a diagram may have reduced this shortcoming, but it led to inflexibility and was confusing to most of the subjects. The flexibility to quickly switch from schema dictionaries to diagrams, and to produce diagrams of evolving schemas was welcomed by many subjects.

4. Further Work.

The prototypes described in this thesis have ample room for improvement or further investigation. We suggest below some directions of further research.

a. Investigating the symbiotic relationship between process and data models.

Investigating the symbiotic relationship between process and data models in the context of schema integration might constitute an interesting research

avenue. [Dinkhoff et al 1994] describe an approach to business process management which is based on modelling data-related, activity-related and organisational aspects of business processes. [Winter 1994]'s work on extending conceptual models to capture basic behavioural system elements (e.g. propagation of deletions, insertions and updates) provides a basis for such work. The suggested research can incorporate ideas from these works.

b. Generalisation and Aggregation Hierarchies.

Generalisation and aggregation are powerful modelling constructs. However, integrating schemas that contain these abstraction mechanisms presents both management and practical problems. Abstraction hierarchies hide a lot of information up/down the hierarchy, resulting in valuable semantic information being readily unavailable to integrators. This problem needs further investigation, in particular the hierarchy-wide implications of integrations involving concepts participating in abstraction relationships. Such an investigation may be extended to the timing of utilising generalisation and aggregation as modelling constructs during schema integration. For now we suggest it is best to utilise them last i.e. during restructuring the final integrated schema. For instance, [Jaeschke et al 1994] widely use these to derive complex object types for business process modelling. [Dupont 1994] goes further by proposing techniques applicable to resolving conflicts between concepts with and without generalisation.

c. Flexibility in Issue-base Maintenance.

It is difficult or even impossible to foresee the integrated schema exhibiting the most desirable characteristics (e.g. communicability, readability). It is therefore desirable to have several versions of an integrated schema involving the same subschemas with a provision to reuse knowledge emerging from previous integration tasks. Data analysts therefore need the ability to quickly and flexibly modify or submit new issues and proposals for integration. SISIBIS's usefulness would be greatly increased if such modifications are propagated across the schema integration hyperdocuments. This is a versioning problem that demands further research. Such work may follow the proposals put forward by [Baxter 1992] who argues *'Updating the design information is as important as revising the software itself, for the revised design information is necessary for further maintenance'*. Baxter proposes a model involving transformation specification, planning, derivation histories and maintenance deltas (e.g. codification of maintenance changes and

classifying them as *perfective, adaptive* and *corrective*). However, it may be misleading to conceive of an integration task as perfective or corrective as the integrated concepts are socially-constructed via negotiation. More recently, [Liu et al 1994] discussed an approach to schema evolution through changes to the E-R schema, effected through EVER (their EVolutionary ER diagram for specifying relationships between schema versions and relationships among attributes.

d. Assessment of socio-technical interaction.

Our work on SISIBIS has attempted to demonstrate the notion that *'Computer science ... should not drive a wedge between the social and the technical, but rather link both throughout the formal and informal ...'* [Friedman and Khan 1994]. SISIBIS supports both the technical and collaborative aspects of schema integration by:

i. systematically supporting a method (the technical aspect) for schema integration. The integration of concepts is formally defined in terms of assertions which include 'why' and 'how' the concepts are integrable.
ii. flexibly supporting collaboration (the social aspect) during schema integration.

However, the model of work used gives no indication of the impact of these aspects on each other. The impact of SISIBIS on the social, political and motivational factors within and across development teams needs thorough investigation. For instance, we would want to assess the extent to which SISIBIS 'naturally' augments participant's native concepts and thinking patterns (assessment of intrusiveness or disruptiveness). More importantly, we might be concerned with the knowledge produced, conveyed and consumed during the integration process with a view to streamline or extend the coverage of SISIBIS.

e. The formal specification of annotations.

It would be interesting to investigate the formal specification of annotations (explanations, comments etc.). For instance, one would define a set of keywords representing conditions, exceptions etc. (e.g. (X,equal,Y) for (condition, X ≤ X Value)). Such specifications, when part of the rationale for an integration decision and linked to the integrated conceptual schema are likely to be useful during coding. By linking the integrated concept, its rationale

(possibly including exceptions and conditions) and source/executable code that refers to it in an environment such as Dynamic Design [Bigelow 1988], developers may be automatically notified of exceptions and results of queries to the resultant database can be further annotated in a spirit similar to that suggested by [Motro 1992]. Additionally, such formalisms would provide opportunities for a 'schema-integration spreadsheet'. The DBA or participating data analysts would enter values for a concept participating in an integration proposal, the spreadsheet would provide the corresponding values (and how they were computed) of the equivalenced concept(s). Thus such a spreadsheet acts as an individual decision-support tool allowing participants to test out their integration proposals before proposing them thus improving the quality of proposals and also instilling a sense of confidence in the participants. This idea, due to [Bieber 1992], is hampered by the static nature of present-day hypermedia authoring systems. Formal specification such as the one proposed above would provide a dynamic extension to the node and linking mechanisms of hyperdocuments.

f. Investigating HSMS-SISIBIS as a Mutual Learning Environment.

Further work may investigate (over a long period) the utility of HSMS-SISIBIS as a learning environment for students studying databases. Inevitably, given the same database design task, students are likely to produce different schemas. A collaborative schema integration exercise (possibly including the lecturer) is an opportune time for students to (mutually) learn modelling techniques as well the underlying semantics of data model constructs. Since the basic structure of conceptual schemas is the same for database schemas, an extension to such work might involve the forward- and backward-engineering of database schemas to reach a normal form [Kent 1983] chosen by the student. (A database schema is said to be in a given normal form if it satisfies certain criteria that reduce anomalies (e.g. redundancy) in the resultant database)

g. Agenda Setting, Negotiation and Focusing: Access-Rights Mechanisms for Nodes.

The process of collaborative schema integration can be considered as 'virtual' (virtual because they do not actually implement the integrations) coauthoring of the conceptual model by participating data analysts. However, it may be desirable for the DBA to have mechanisms for agenda setting, negotiation and focusing. Establishing links to a system offering such facilities would be useful. This requires better file sharing facilities. As we have seen,

the local-copy file sharing option proved inadequate as it compromised performance. Ideally, a system for agenda setting, negotiation and focusing would support real-time on-line file sharing (i.e. there is only one file). Further, the system would require node access mechanisms to determine who to and when to give access rights to a particular node (e.g. card, field, button etc.). Developing a hypermedia authoring system offering such facilities seems a worthy research project.

5. Conclusion.

The applicability of hypermedia to database design has been investigated and discussed in this thesis. We have provided a theoretical perspective for hypermedia, schema modelling and (collaborative) schema integration. Further, discussion has been provided of working prototypes that support schema modelling and collaborative schema integration in a hypermedia environment.

The main themes emerging from the work are that:

a. hypermedia appears suitable for the representation and management of the dictionary components of conceptual modelling documents. However, HyperCard as a representative of hypermedia technology, provides inadequate support for the representation of large diagrams. Import/export facilities from/to more flexible diagramming tools would alleviate this problem.
b. a hypermedia-based argumentation framework such as the IBIS scheme positively supports the collaborative nature of schema integration. The quality of schemas resulting from such a framework is likely to be good because concept integrations are based on semantic consensus and the rationale for concept integration decisions is captured. Further, such a framework is a mutual learning environment in which data analysts (new and old) can understand the semantics of organisational data.

The future of hypermedia is promising. As IS departments strive to manage the software backlog, considerable time is needlessly spent trying to regenerate or rediscover the rationale for certain design decisions. Specifically, in the field of database design where accurate data can mean the difference between success and failure, and where reality is socially-constructed using tools (data models) that are not currently rich enough, we need novel ways of correctly representing the semantics of business data. We foresee hypermedia, and

particularly problem-resolution systems, playing a central role in CAISE environments of the future.

Evaluation Questionnaire.

Name: **Date:** **SISIBIS Questionairre LBQ1a.**

Q1. How would you rate your knowledge of E-R modelling?
a. Poor b. Good c. Very good

Q2. How well do you understand the concepts of generalisation and aggregation?
a. Poor b. Good c. Very good

Q3. Did you have prior knowledge of schema integration prior to this exercise?
a. Yes b. No

Q4. If you answer to question Q3 is Yes, how would you rate this knowledge?
a. Poor b. Good c. Very good

Q5. Did you find the need to talk to other members of your group?
a. Yes b. No c. Don't know

Q6. If your answer to Q5 is Yes:
Q6.1 briefly specify what you needed to talk about.

Q6.2 did you find it useful talking to them?
a. Yes b. No c. Don't know

Q7. How would you rate the reasons put forward for the integration proposals by other members of your group:
a. poor b. good c. very good.

Q8. What features in the system did you find most useful?

Q9. What features did you find lacking in the system?

Q10. Suggest ways in which the demonstrator may be improved?

Q11. Would you be willing to participate in further sessions in future?
a. Yes b. No

Q12. Please specify any further comments you wish to make about the exercise?

Thank You for Your Cooperation.

Evaluation Session GuideLines.

1. Description of Domain.

GlamPublishing is a hypothetical department that has been established to oversee all the research in the university and its colleges. In particular it is tasked with the management and marketing of research documents produced in the university. The documents produced include open learning material (research newsletters, papers, books) authored at the university. The ultimate aim of GlamPublishing is to become commercial. It is envisaged that a number of publishing companies will be tasked with printing the documents. On receipt of the documents, the marketing arm of the department will sell some of the material to students on the various courses in the colleges. They have also been tasked with tracking the courses that use the books authored by the university's lecturers as the main text. Additionally, the publications may be loaned to students taking a course that uses the publication as the main text.

To effectively keep track of the published research documents in the university, the IT Centre has been tasked with producing a structure for a database to manage the documents. The head of the centre has decided to assign two data analysts, A and B, to this job. Instead of assigning different areas to model to each of the analysts, it has however been decided that they independently produce models of the same area. These models (schemas) are then compared and contrasted, and then merged to produce a unified schema. Further it has been decided that the design will be undertaken according to the following likely order of business objectives:

a. The identification of the major research areas relevant to the documents.
b. The authorship of research documents.
c. Publication of the documents.
d. The marketing of the publications to other universities and colleges.
e. The adoption of books as main texts for courses and the loaning of publications to students on various courses in the colleges.

Data analysts A and B were each expected to produce 5 partial schemas, each corresponding to the items (a) - (e) above (Figure 10.1 (a) - (e) below).

2. Integration Tasks.

Assuming analyst A and B are part of your team, you are tasked with producing the integrated schema for each of the pairs of schemas indicated in Figures 10.1(a) to (e) below using the tool provided.

Figure 10.1 (a): Research Areas Schemas.

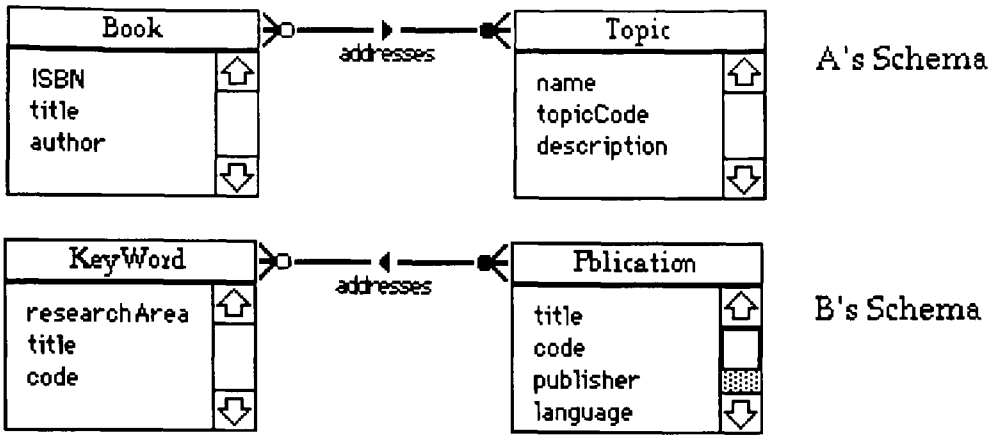

Figure 10.1 (b): Authoring of Documents Schemas.

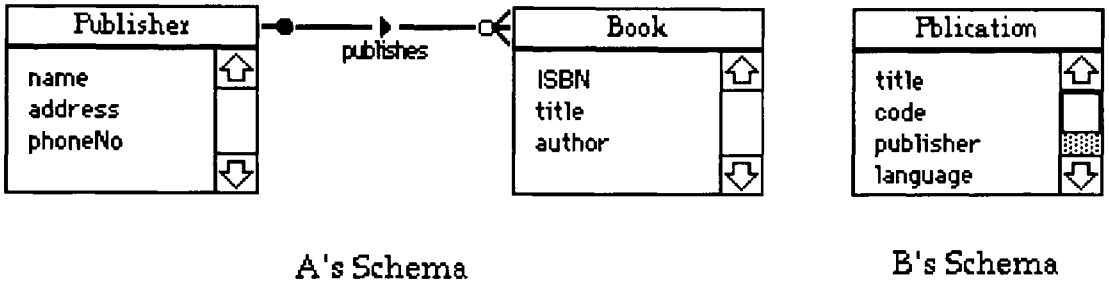

Figure 10.1 (c): Printing of Documents Schemas.

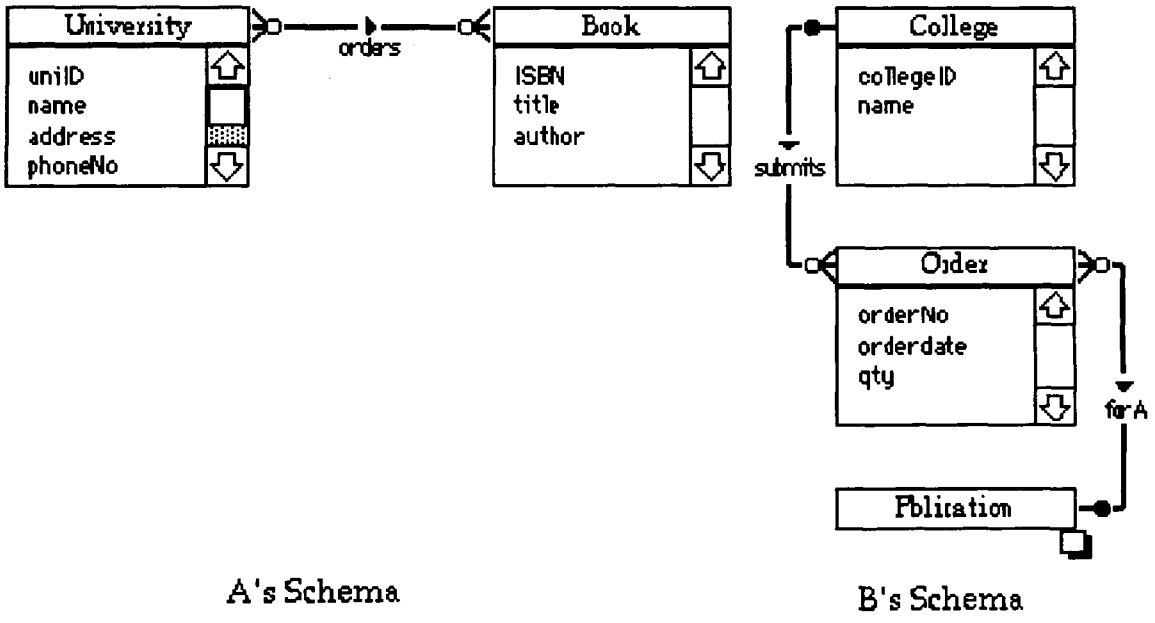

Figure 10.1 (d): Ordering of Documents Schemas.

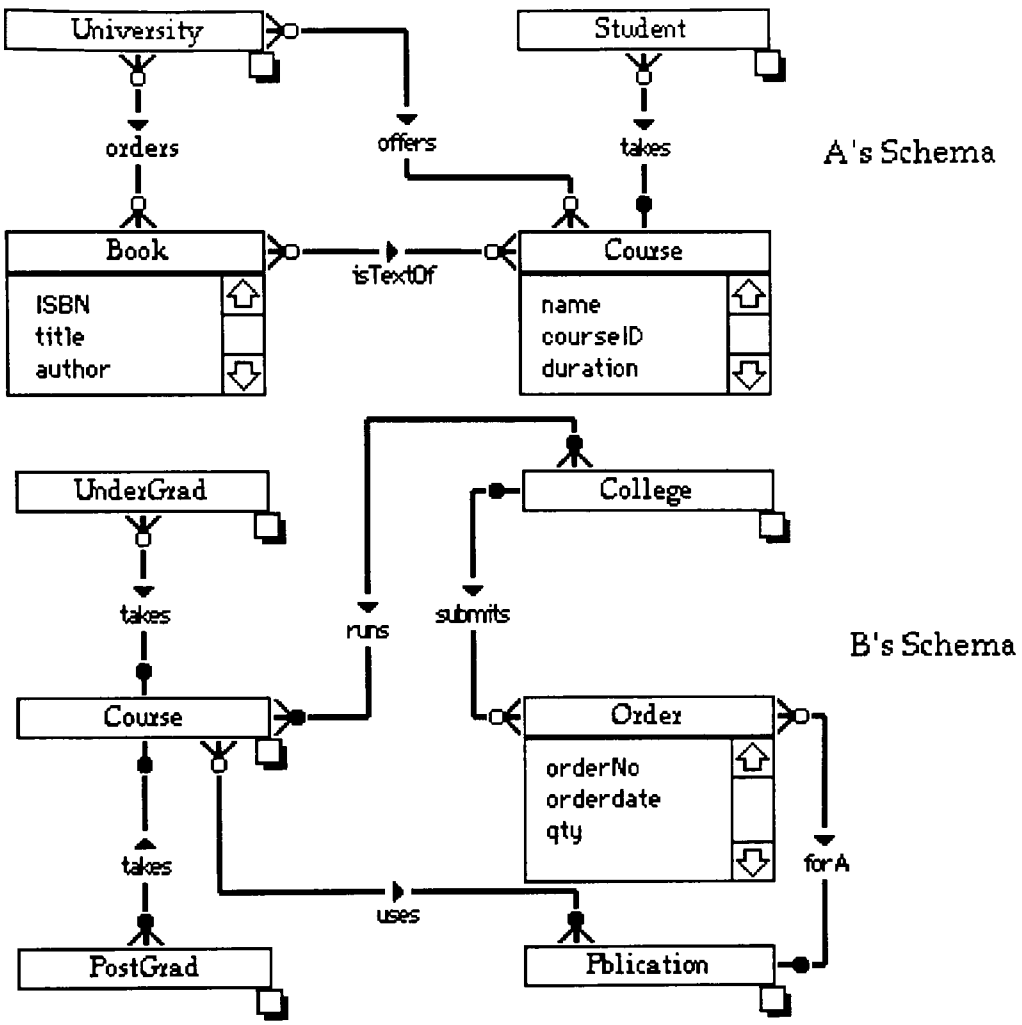

Figure 10.1 (e): Marketing of Documents Schemas.

3. Example of a Schema Integration Task.

The schemas shown in Figure 10.2 represent subsets of the Birth Registry and the National Registry respectively. For the sake of argument, let's assume these schemas are to be integrated so that a child born is automatically registered as a national.

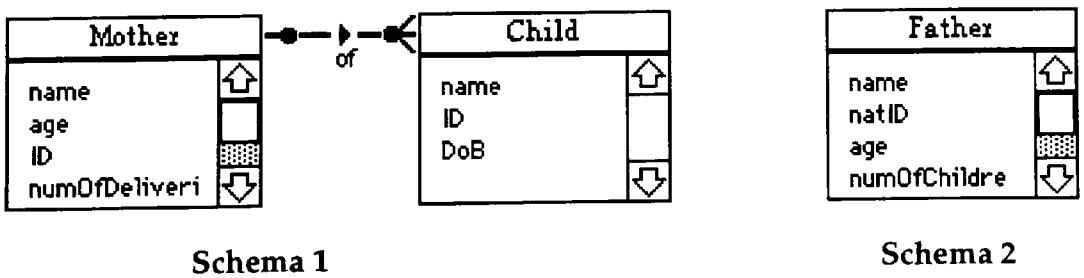

Figure 10.2: Sample Schemas for Birth and National Registry to be Integrated.

On examination of the two schemas, the following observations may be made (where ≈ represents equivalence):

a. (Mother.name ≈ Father.name) *i.e. name is a word used to identify a person.*
b. (Mother.natID ≈ Fathe.ID) *i.e. the national insurance number uniquely identifies a person.*
c. (Child.DoB ≈ Mother.age) *i.e. a person's DoB can be used to calculate their age.*
d. (Mother.NumOfDeliveries ≈ Father.NumOfChildren). *However, the birth of twins per delivery makes this assumption invalid, so these attributes are left alone. Similarly, an inaccurate assumption about a child's unique birth ID and the mother's (or father's) national IDs may be made. Usually this kind of knowledge resides in the minds of the modellers of the schemas.*

The main function of the tool is thus to allow integration teams to define such assertions and have other members of the team 'validate' them via entering responses in the form of explanations, comments, questions, answers and examples of actual data instances that substantiate an annotation.

Figure 10.3 shows the final integrated schema of the subschema examples just discussed. The entity 'Parent' generalises the entities 'Mother' and 'Father', while the entity 'Parent' is associated with the entity 'Child' via the relationship 'of'. i.e. a child belongs to both parents. However, it is the duty of the leader of your group to make the final decision and implement the mergers.

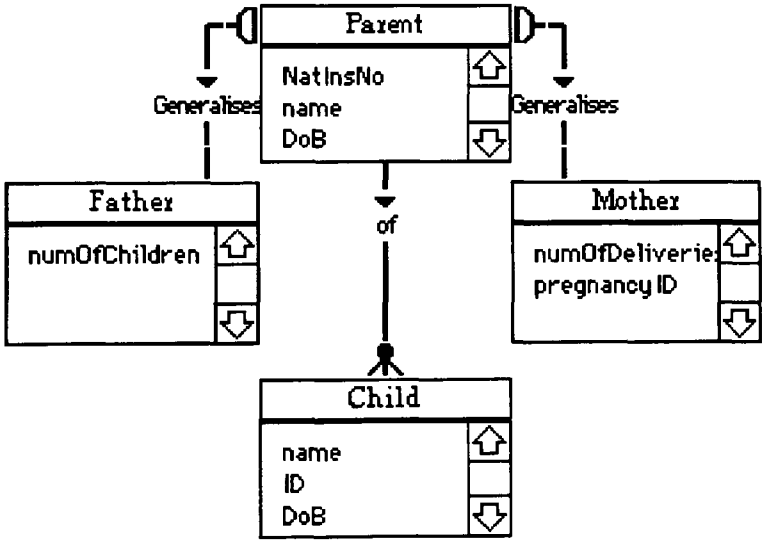

Figure 10.3: Possible Final Integrated Schema of Registration Example.

During the exercise you are therefore asked to examine:

a. the schemas for equivalences and define your assertions,
b. the assertions entered by members of your group and respond to them if you feel there is a need to.

Your cooperation in this exercise is greatly appreciated.

List of References.

Agosti M. and Johnson R. G., 1984, "A Framework of Reference for Database Design", *DATA BASE*, Summer 1984, pp 3 - 9.

Alschuler L., 1989, "Hand-Crafted Hypertext- Lessons from the ACM Experiment", The Society of Text, E. Barret (ed.), MIT Press.

Apple Computer Inc., 1990, "HyperCard User's Guide", Cupertino, CA.

Applegate L., 1991, "Technology Support for Cooperative Work: A Framework for Studying Introduction and Assimilation in Organisations", *Journal of Organisational Computing*, 1, pp 1 - 10.

Atzeni P., Batini P., Antonellis V., Lenzerini V., Villanelli F. and Zonta B., 1982, "A Computer-Aided Tool for Conceptual Database Design", *Proc. of IFIP Conference onAutomated Tools for Information Systems Design*, Schneider H. and Wassermann A. (eds), North-Holland Publishing Company, IFIP, Amsterdam, pp 85 - 106.

Avison D. E, 1992, "Information Systems Development: A database approach" (2nd Edition), Oxford, Blackwell Scientific.

Bachman C. W., 1969, "Data Structure Diagrams", *DATA BASE , 1[2]*, pp 4 - 10.

Barclay P. J. and Kennedy J. B., 1991, "Regaining the Conceptual Level in Object-Oriented Modelling", Jackson M. S. and Robinson A. E. (eds), Aspects of Databases, *Proc. of the 9th British National Conference on Databases*, Butterworth-Heinemann, pp 269 - 305.

Baskerville R., 1993, "Semantic Database Prototypes", *Journal of Information Systems [3]*, pp 119 - 144.

Batini C. and Lenzerini M., 1984, "A Methodology for Data Schema Integration in the Entity-Relationship Model", *IEEE Transaction on Software Engineering, 10[6]*, pp 650 - 664.

Batini C., Lenzerini M. and Moscarini M., 1983, "Views Integration", Methodology and Tools for Data Base Design, Ceri(ed), pp 57 - 84.

Batini C., Lenzerini M. and Navathe S., 1986, "A Comparative Analysis of Methodologies for Database Schema Integration", *ACM Computing Surveys, 18[4]*, 323 - 364.

Baxter I. D., 1992, "Design maintenance Systems", *Communications of the ACM, 35 [4]*, pp 73 - 89.

Begeman M. L. and Conklin J., 1988, "The Right Tool for the Job", *BYTE, October*, pp 255 - 266.

Bell P. and Evans, 1989, "Mastering Documentation", Wiley, U.S.A.

Bernstein M., Bolter J., Joyce M. and Mylonas E., 1991, "Architectures for Volatile Hypertext", *Proc. of Hypertext '91*, pp 243-260.

Bernstein P. A., 1976, "Synthesising Third Normal Form Relations from Functional Dependencies", *ACM Transactions on Database Systems, 1[4]*, pp 277 - 298.

Bersoff E. H. and Davis A. M., 1991, "Impact of Life-Cycle Models on Software", *Communications of the ACM, 34 [8]*, pp 104 - 118.

Beynon-Davies P, 1993, "Information Systems Development" (Second Edition), Macmillan, London.

Beynon-Davies P., 1992 (a), "Entity Models to Object Models: Object-Oriented Analysis and Database Design", *Information and Software Technology, 34[4]*, pp 255 - 261.

Beynon-Davies P., 1992 (b), "The Realities of Database Design: an essay on the sociology, semiology and pedagogy of database work", *Journal of Information Systems*, 2, pp 207 - 220.

Bieber M., 1992, "Automating Hypermedia for Decision Support", *Hypermedia, 4 [2]*, pp 83 - 110.

Bigelow J., 1988, "Hypertext and CASE", *IEEE Software, 5 [2]*, pp 23 - 27.

Bishop T., 1988, "The Software Development Life Cycle", *.EXE Magazine, 2[9]*, pp 40 - 43.

Blaha M., Premerlani W. J. and Rumbaugh J. E., 1988, "Relational Database Design Using An Object-Oriented Methodology", *Communications of ACM, 23[4]*, pp 414 - 427.

Botafogo R. and Shneiderman B., 1991, "Identifying Aggregates in Hypertext Structures", *Proc. of Hypertext '91*, pp 63-73.

Bouzeghoub M. and Comyn-Wattiau I., 1990, "View Integration By Semantic Unification and Transformation of Data Structures'", *Proc. of the 9th International Conference on Entity-Relationship Approach*, Lausanne, 1990, pp 413 - 430.

Brodie M. L., 1984, "On The Development of Data Models", in Brodie M. L., Mylopoulos J. and Schmidt T. W. (eds), On Conceptual Modelling: perspectives from artificial intelligence, databases and programming languages, Springer-Verlag.

Brown J. and Duguid P., 1994, "Borderline Issues: Social and Material Aspects of Design", *Human-Computer Interaction, 9*, pp 3 - 36.

Burns M. and Whitten W. B. (II), 1990, "A Hypertext Database For User Interface Design", *Proc. of the 13th International Conference on Human Factors in Telecommunications*, pp 7 - 13.

Bush, V., 1945, "As We May Think", *Atlantic Monthly*, July, pp 101-108

Butler J., 1987, "What the Leaders are Saying?", *System Development*, November, pp 6 - 9.

Byrant T., 1989, "Structured Methodologies and Formal Notations: developing a framework for synthesis and investigation", *Z User Workshop*, Oxford, Springer-Verlag.

Ceri S. and Pelagatti G., 1984, "Distributed Databases: Principles and Systems", McGraw-Hill, Singapore.

Chen M. and Nunamaker J. F., 1991, "The Architecture and Design of A Collaborative Environment for System Definition'", *DATA BASE*, Winter/Spring, pp 22 - 29.

Chen M., Nunamaker J. and Weber E.S., 1989, "Computer-Aided Software Engineering: Present and Future Directions", *DATA BASE, 20[1]*, pp 7 - 13.

Chen P., 1976, "The Entity-Relationship Model- Toward a Unified View of Data", *ACM Transactions on Database Systems, 1 [1]*, pp 9 - 36.

Christodoulakis S., Theodoridou M., Ho F., Papa M. and Pathria A., 1986, "Multimedia Document Presentation, Information Extraction, and Document Formation in MINOS: A Model and a System", *ACM Transactions on Office Information Systems, 4 [4]*, pp 345 - 383.

Codd E. F., 1970, "A Relational Model of Data for Large Shared Data Banks", *Communications of the ACM, 13[6]*, pp 377 - 387.

Codd E. F., 1979, "Extending the Database Relational Model to Capture More Meaning", *ACM Transactions on Database Systems, 4[4]*, pp 397 - 434.

Collet C., Huhns M. N. and Shen W-M., 1991, "Resource Integration Using a Large Knowledge Base in Carnot", *IEEE Computer*, pp 55 - 62.

Computer Systems Advisor Inc., -, "Silverun", Woodcliff Lake, New Jersey.

Conklin J. and Begeman M. L., 1988, "'gIBIS: A hypertext tool for explolatory policy discussion'", *ACM Transactions on Office Information Systems, 6[4]*, pp 303 - 331.

Conklin J., 1987, "Hypertext: An Introduction and Survey", *IEEE Computer 20 [9]*, pp 17 - 41.

Cybulski J. and Reed K., 1992, "A Hypertext Based Software Engineering Environment", *IEEE Software, 9 [2]*, pp 62 - 68.

Czejdo B., Elmasri R., Rusinkiewicz M. and Embley D. W., 1990, "A Graphical Data Manipulation Language for an Extended Entity-relationship Model", *IEEE Computer, 23[3],* pp 26 - 36.

Date C. J., 1990, "An Introduction to Database Systems", Volume 1 (5th Edition), Addison Wesley Publishing Company.

Delisle N. and Schwartz M., 1986, "Neptune: A Hypertext System for CAD Applications", *Proc. of the ACM SIGMOD '86 International Conference on Management of Data,* Washington DC., pp 132 - 143.

Delisle N. and Schwartz M., 1987, "Contexts: A Partioning Concept for Hypertext", *ACM Transactions on Office Information Systems, 5[2],* pp 168 - 186.

Desai B. C., 1990, "An Introduction to Database Systems", West Publishing Company, USA.

Dewan R. M. and Gavish B., 1989, "Models for the Combined Logical and Physical Design of Databases", *IEEE Transactions on Computers, 38 [7],* pp 955 - 967.

Dillon, A., McKnight, C. and Richardson, J., 1990, "Navigation in Hypertext: A critical review of the concept", *Proc. of Interact '90,* pp 587 - 591

Dinkhoff G., Gruhn V. Saalmann A. and Zielonka M., 1994, "Bussiness Process Modelling in the Workflow Management Environment **Leu**", *Proc. of the 13th International Conference on the Entity-Relationship Approach,* Manchester (U.K.), pp 46- 63.

Dupont Y., 1994, "Resolving Fragmentation Conflicts in Schema Integration", *Proc. of the 13th International Conference on the Entity-Relationship Approach,* Manchester (U.K.), pp 513 - 532.

Eick C. F. and Lockemann P. C., 1985, "Acquisition of Terminological Knowledge Using Database Design Techniques", *ACM Proc. of the International Conference on Management of Data* , Austin (Texas), pp 84 - 94.

Ellis H., 1982, "A Refined Model for the Definition of Systems Requirements", *Database Journal, 12[8]*, pp 2 - 9.

Elmasri R. and Navathe S., 1984, "Object Integration in Logical Database Design", *Proc. of the 1st International Conference on Data Engineering*, Los Angeles, pp 426 - 433.

Elmasri R. and Navathe S., 1989, "Fundamentals of Database Systems", Addison-Wesley.

Elmasri R., Larson J. and Navathe S. B., 1987, "Integration Algorithms for Federated Databases and Logical Database Design", *Technical Report*, Honeywell Corporate Research Center.

Engelbart D. C. and English W. C., 1968, "A Research Center for Augmenting Human Intellect", *American Federation of Information Processing Societies Conference Proc. of the Fall Joint Computer Conference*, 33, Thompson Book Company, Washington DC, pp 395 - 410.

Engelbart D. C., 1963, "A Conceptual Framework for the Augmentation of Man's Intellect", *Vistas of Information Handling, 1*, pp 1 - 29, Spartan Books, London.

Engelbart D. C., 1975, "NLS Teleconferencing Features: The Journal, and Shared Screen Telephoning", *Digest of papers, IEEE Computer Society Conference (CompCon75)*, pp 173 - 176.

Excel Software, -, "MacAnalyst/Combo", Marshalltown, IA.

Farollon Computing Inc., 1990, "Farollon SoundEdit 2.0 User's Guide", Emeryville, Carlifornia, USA.

Feldman P. and Miller D., 1986, "Entity model clustering: Structuring A Data Model By Abstraction", *The Computer Journal, 29 [4]*, pp 348 - 360.

Fiderio J., 1988, "A Grand Vision: Hypertext mimics the brain's ability to access information quickly and intuitively by reference", *BYTE, October*, pp 237 - 244.

Fischer G., Grudin J., Lemke A., McCall R., Ostwald J., Reeves B. and Shipman F., 1992, "Supporting Indirect Collaborative Design With Integrated Knowledge-Based Design Environments", *Human-Computer Interaction, 7*, pp 281 - 314.

Fischer G., Lemke A., Mastaglio T. and Morch A., 1991(a), "The Role of Critiquing in Cooperative Problem Solving", *ACM Transactions on Office Information Systems, 9[2]*, pp 123 - 151.

Fischer G., Lemke A., McCall. R and Morch A., 1991(b), "Making Argumentation Serve Design", *Human-Computer Interaction, 6*, pp 393 - 419.

Flynn D., 1992, "Information Systems Requirements: Determination and Analysis", McGraw-Hill, Maidenhead, Berks, U. K.

Fogel S., 1992, "Data Modelling with Macintosh CASE Tools", *Database Programming and Design, November*, pp 43 - 48.

Friedman B. and Khan P. H. jnr, 1994, "Educating Computer Scientists: Linking the Social and the Technical", *Communications of the ACM, 37 [1]*, pp 65 - 70.

Frisse M. and Cousins, S., 1989, "Information Retrieval form Hypertext: Update on the Dynamic Medical Handbook", *Proc. of Hypertext '89*, pp 199-211.

Garg P. K. and Scacchi W., 1988, "A Hypertext System To Manage Software Life-Cycle Documents", *Proc. of the 21st Hawaii International Conference on System Science, 2*, IEEE Computer Society Press, pp 337 - 345.

Garg, P., 1988, "Abstraction Mechanisms in Hypertext", *Communications of the ACM, 31[7]*, pp 862-870.

Gloor P., 1991, "Cybermap: Yet Another Way of Navigating in Hyperspace", *Proc. of Hypertext '91*, pp 107-121.

Goodman D., 1990, "The Complete HyperCard Handbook", Bantam Books, New York.

Gordon M. D. and Fry J. P., 1989, "Novel Application of Information Retrieval to the Storage and Management of Computer Models", *Information Processing and Management, 25[6]*, pp 629 - 646.

Greif I., 1988, "Computer-Supported Cooperative Work: A Book of Readings", San Mateo, CA, Morgan Kaufmann.

Gronbaek K., Kyng M. and Mogensen P., 1993, "CSCW Challenges: Cooperative Design in Engineering Projects", *Communications of the ACM, 36 [4]*, pp 67 - 77.

Grudin J., 1990, "Groupware and Cooperative Work: Problems and Prospects", *The Art of Human-Computer Interface Design*, Laurel (ed), Reading, MA, Addison-Wesley, pp 171 - 185.

Grudin J., 1994, "Groupware and Social Dynamics: Eight Challenges For Developers", *Communications of the ACM, 37 [1]*, pp 92 - 105.

Halasz F., 1988, "Reflections on NoteCards: Seven issues for the next generation of Hypermedia Systems", *Communications of the ACM, 31[7]*, pp 836-851.

Hammer M. and McLeod D., 1981, "Database description with SDM: A Semantic Database Model", *ACM Transactions on Database Systems, 6[3]*, 351 - 386.

Hara Y., Keller A. and Wiederhold G., 1991, "Implementing Hypertext Database Relationships through Aggregations and Exceptions", *Proc. of Hypertext '91*, pp 75-90.

Harriman C., 1987, "Owl International's Guide: hypertext comes to the small screen", *Seybold Outlook on Professional Computing, 5[7]*, pp 14 - 18.

Hayne S. and Ram S., 1990, "Multi-User View Integration System (MUVIS): An Expert System for View Integration", *6th International Conference on Data Engineering*, Los Angeles, Carlifornia (IEEE Computer Society), pp 402 - 409.

Howe G. A., 1993, "A Collision of Semantics", *Database Programming and Design, February*, pp 54 - 61.

Hsu C., Bouziane M., Rattner L. and Yee L., 1991, "Information Resource Management in Heterogeneous, Distributed Environments: A Metadatabase Approach", *IEEE Transactions on Software Engineering, 17 [6]*, pp 604 - 625.

Iconix Software Engineering Inc., -, "DataModeller", Santa Monica, CA 90405.

Index Technology Corp., 1987, "Excelerator", *Proc. of CASE Symposium on Digital Consulting.*

Irish P. M. and Trigg R. H., 1989, "Supporting Collaboration in Hypermedia: Issues and Experience", *Journal of American Society of Information Science, 40[3]*, pp 192 - 199.

Ishii H., 1991, "TeamworkStation: Towards A Seamless Shared Workspace", *ACM Proc. of the Third International Conference on Computer-Supported Cooperative Work (CSCW '90)*, pp 13 - 26.

Jaeschke P., Oberweis A. and Stucky W., 1994, "Deriving Complex Structured Objects Types fro Bussiness Process Modelling", *Proc. of the 13th International Conference on the Entity-Relationship Approach*, Manchester (U.K.), pp 28 - 45.

Jajodia S., NG P. A. and Springsteel F. N., 1983, "The problem of Equivalence for Entity-Relationship Diagrams", *IEEE Software Engineering, 9[5]*, pp 617 - 630.

Jones M. R., 1992, "Unveiling Repository Technology", *Database Programming and Design, April*, pp 28 - 35.

Kambanis J., 1990, "A Design Environment for Semantic Data Models", *SART/SACJ, 1*, pp 24 - 30.

Kent W., 1983, "A Simple Guide to Five Normal Forms in Relational Database Theory", *Communications of the ACM, 26[2]*, pp 120 - 125.

Kent W., 1991, "The Breakdown of the Information Model in Multi-Database Systems", *SIGMOD Record, 20 [4]*, pp 10 - 13.

Knight J. C. and Myers E. A., 1993, "An Improved Inspection Technique", *Communications of the ACM, 36 [11]*, pp 50 - 61.

Kotteman J., Gordon M. and Stott J. W., 1991, "A Storage and Access Manager for Ill-Structured Data", *Communications of the ACM, 34 [8]*, pp 94 - 103.

Larson J. A., Navathe S. B. and Elmasri N., 1989, "A Theory of Attribute Equivalence in Databases with Application to Schema Integration", *IEEE Transaction on Software Engineering, 15[4]*, pp 449 - 463.

Laurel B., 1991, "Interface Agents: Metaphors with Character", *The Art of Human Computer Interface Design*, Addison Wesley Reading Mass., pp 355 - 366

Leite J. C. S. and Freeman P. A., 1991, "Requirements Validation Through Viewpoint Resolution", *IEEE Transactions on Software Engineering, 17[12]*, pp 1253 - 1269.

Lenat D. and Guha R. V., 1990., "Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project", Addison-Wesley, Reading, Mass.

Leonard M., 1992, "Database Design Theory", Macmillan Education, Basingstoke, Hampshire, England, U.K.

Liu C-T., Chrysanthis P. K. and Chang S-K., 1994, "Database Schema Evalution through the Specification and Maintenance of Changes on Entities and Relationships", *Proc. of the 13th International Conference on the Entity-Relationship Approach*, Manchester (U.K.), pp 132 - 151.

Logic Works Inc., -, "ERwin/ERX", Princeton, New Jersey.

Loomis M. E. S., Shah A. V. S. and Rumbaugh J. E., 1987, "An Object-Oriented Modelling Technique for Conceptual Design", *Proc. of the European Conference on Object-Oriented programming*, pp 121 - 140.

Lundberg B., 1982, "An Axiomatization of Events", *BIT, 22*, pp 291 - 299

Lundenberg A., 1982, "Information Modelling Tool", Automated Tools for Information Systems Design, Schneider H. J. and Wasserman A. I. (editors), North-Holland Publishers.

Malone T. W. and Lai K., 1988, "Object Lens: A Spreadsheet for Cooperative Work", *Proc. of CSCW '88*, Portland, Oregon, U.S.A.

Martin C. F., 1988, "Second Generation CASE tools: A challenge to vendors", *IEEE Software, 5[2]*, pp 46 - 49.

Martin J. and McLure C., 1985, "Diagramming Techniques for Analysts and Programmers", Prentice-Hall, Englewood Cliffs, New Jersey.

Martin J., 1989, "Information Engineering", Prentice-Hall, Books 1 - 3.

Mayes J. T., Kibby M. and Watson H., 1988, ""StrathTutor: the development and evaluation of a learning-by-browsing system on the Macintosh", *Computer in Education, 12*, pp 221 - 229.

McCall R. J., 1988, "PHI: A Hypertext Approach to Handling Design Information", *7th International Conference on Information for Designers*, (Southampton, U.K), pp 1 - 18.

McCall R. J., 1991, "PHI: A Conceptual Foundation For Design Hypermedia", *Design Studies, 12[1]*,.

McDermid J., 1985, "Integrated Project Support Environments", Peter Peregrinus, London.

Miller G.A., 1956, "The magical number seven ±2: some limits on our capacity for processing information", *Psychological Reviews, 63 [2]*, pp 81 - 96.

Mintzberg H., Raisinghani D. and Thoeret A., 1976, "The Structure of Unstructured Decision Processes", *Administrative Science Quarterly, 21*.

Moody D. L. and Shanks G. G., 1994, "What Makes a Good Data Model? Evaluating the Quality of Entity-Relationship Models", *Proc. of the 13th International Conference on the Entity-Relationship Approach,* Manchester (U.K.), pp 94- 111.

Moriarty T., 1992(a), "Architecture for change with attributes involves some important decisions", *Database Programming and Design, June,* pp 65 - 67.

Moriarty T., 1992(b), "Change management: What part does the repository play?", *Database Programming and Design, August,* pp 65 - 67.

Motro A., 1987, "Superviews: Virtual Integration of Multiple Databases", *IEEE Transactions on Software Engineering, 13 [7],* pp 785 - 799.

Motro A., 1992, "Annotating Answers with Their Properties", *SIGMOD RECORD, 21[1],* pp 54 - 57.

Mylopoulos J. and Schmidt T. W. (eds), 1984, "On Conceptual Modelling: perspectives from artificial intelligence, databases and programming languages", Springer-Verlag.

Navathe S, Elmasri R. and Larson J., 1986, "Integrating User Views in Database Design", *IEEE Computer, 19[1],* pp 50 - 62.

Nelson T., 1965, "The Hypertext", *Proc. of the International Documentation Federation.*

Nelson T., 1987, "Literary Machines", 87.1 edition, Project Xanadu, 8480 Fredericksburg #138, San Antonio.

Nielsen J., 1993, "Noncommand User Interfaces", *Communications of the ACM, 36[4],* pp 83 - 99.

Niemi T. and Jarvelin K., 1991, "Prolog-Based Meta-Rules for Relational Database Representation and Manipulation", *IEEE Transactions on Software Engineering, 17 [8],* pp 762 - 788.

Olerup A., 1991, "Design Approaches: A Comparative Study of Information System Design and Architectural Design", *The Computer Journal, 34[3]*, pp 215 - 224.

Olson G. M. and Olson J. S., 1991, "User-Centred Design of Collaboration Technology", *Journal of Organisational Computing, 1*, pp 61 - 83.

Petroski H., 1985, "To Engineer Is Human: The Role of Failure in Successful Design", St. Martin's Press, New York.

Rada R., 1991, "Hypertext: from text to expertext", McGraw-Hill Book Company (UK) Limited, U.K.

Reiner D., Brodie M., Brown G., Friedell, Kramlich D., Lehman J. and Rosenthal A., 1984, "The Database Design and Evaluation Workbench (DDEW) Project at CCA", *IEEE Database Engineering, 7[4]*, pp 34 - 39.

Rettig M., 1993, "Cooperative Software", *Communications of the ACM, 36[4]*, pp 23 - 28.

Rittel H. and Kunz W., 1970, "Issues as Elements of Information Systems", Center for Planning Development Research, University of California, Berkeley.

Rittel H. and Weber M., 1973, "Dilemmas in a General Theory of Planning", *Policy Sciences, 4.*

Rittel H. W. J., 1984, "Second Generation Design Methods", Developments in Design Methodology, Cross N. (ed), Wiley, New York, pp 317 - 327.

Rittel H., 1971, "Some Principles for the Design of an Educational Information System for Design", *Journal of Architectural Education, 26*, pp 16 - 27.

Rodden T., 1991, "A Survey of CSCW Systems", *Interacting with Computers, 3 [3]*, pp 319 - 353.

Rodden T., Mariani J.A. and Blair G., 1992, "Supporting Cooperative Applications", *Computer Supported Cooperative Work, 1*, pp 41 - 67.

Royce W., 1987, ""Managing the Development of Large Software Systems: Concepts and Techniques"", *9th International Conference on Software Engineering*, Washington D.C., IEEE Computer Society Press, pp 328 - 338.

Salton G., 1989, "Automatic Text Processing: The Transformation Analysis, and Retrieval of Information by Computers", Addison Wesley, Reading, MA.

Saltor F., Castellanos M. and Garcia-Solaco M., 1991, "Suitability of Data Models as Canonical Models for Federated Databases", *ACM SIGMOD RECORD, 20[4]*, pp 44 - 48.

Scacchi W., 1989, "The USC System Factory Project", *SIGSOFT Software Engineering Notes, 14[1]*, pp 61 - 82.

Schmidt K. and Bannon L., 1992, "Taking CSCW Seriously", *Computer Supported Cooperative Work, 1*, pp 7 - 40.

Select Software Tools, 1993, "Select SSADM User Guide", England, U.K.

Sheth A. P., Larson J., Cornelio A. and Shamkant B. N., 1988, "A Tool for Integrating Conceptual Schemas and User Views", *IEEE Proc. of the International Conference on Data Engineering, February*, pp 176 - 183.

Sheth A., 1991, "Semantic Issues in Multidatabase Systems", *ACM SIGMOD RECORD, 20[4]*, pp 5 - 9.

Shipman D., 1981, "The Functional Data Model and the Data Language DAPLEX", *ACM Transactions on Database Systems, 6[1]*, pp 140 - 173.

Shlaer S. and Mellor S. J., 1988, "Object-Oriented Systems Analysis- Modelling the World in Data", Prentice Hall.

Siau K. L., Chan H. C. and Tan K. P., 1992, "A CASE tool for conceptual database design", *Information and Software Technology, 34[12]*, pp 779 - 786.

Sihto M., 1989, "Distributed Hypertext as a Basis for Communication and Collaboration Tools in Distributed Software Environments", *Proc. of the European Conference on Computer-Supported Cooperative Work,*

Silicon Beach Software Inc., 1989, "Aldus SuperCard 2.0", San Diego, CA.

Simon H., 1977, "The New Science of Management Decisions", Harper and Row (New York).

Sjoberg D., 1993, "Quantifying Schema Evolution", *Information and Software Technology, 35 [1]*, pp 35 - 44.

Smith J. M. and Smith D. C. P., 1977, "Database Abstractions: Aggregation and Generalisation", *ACM Transactions on Database Systems, 2*, pp 105 - 133.

Sowa J. F., 1984, "Conceptual Structures: Information Processing in Mind and Machine", Addison-Wessley, Reading, Mass.

Stefik M. and Bobrow D. G., 1986, "Object-Orientated Programming: Themes and Variations", *The AI magazine, January* , pp 40 - 62.

Storey V. C. and Goldstein R. C., 1990, "Design and Development of an Expert Database Design System", *International Journal of Expert Systems, 3[1]*, pp 31 - 63.

Streitz N., Halasz F., Ishii H., Malone T., Neuwirth C. and Olson G., 1991, "The Role of Hypertext for CSCW Applications", *Proc. of Hypertext '91* , pp 369 - 377.

Symantec Corp., 1990, "Think Pascal User Manual", Cupertino, CA.

Sysbase, -, "Deft", Los Angeles, California.

Teorey T. J., Yang D. and Fry J. P., 1986, "A Logical Design Methodology for Relational Databases using the Extended Entity-Relationship Model", *ACM Computing Surveys, 18[2]*, pp 192 - 222.

Tomek I. and Maurer H., 1992, "Helping The User To Select A Link", *Hypermedia, 4[2]*, pp 111 - 122.

Trigg R. H. and Weiser M., 1986, "TEXTNET: A Network-Based Approach to Text Handling, *ACM Transactions on Office Information Systems, 4[1]*, pp 1-23.

Trigg R. H., Suchman L. and Halasz F. G., 1986, "Supporting Collaboration in NoteCards", *Computer Supported Cooperative Work (CSCW '86)*, pp 153 - 162.

Trigg, R., 1988, "Guided Tours and Tabletops: Tools for communicating in a Hypertext Environment", *ACM Transactions on Office Information Systems, 6[4]*, pp 398 - 414

Tsichritzis D. C. and Lochovsky F. N., 1982, "Data Models", Prentice-Hall, Englewood Cliffs, New Jersey.

Twidale M., 1993, "Redressing the balance: the advantages of informal evaluation techniques for Intelligent Learning Environments", *Technical Report*, University of Lancaster (U.K.) (To appear in Journal of Artificial Intelligence In Education).

van Dyke P. H., 1991, "Don't Link Me In: Set Based Hypermedia for Taxonomic Reasoning", *Proc. of Hypertext '91*, pp 233-242.

Vossen G., 1990, "Data Models", Database Languages and Database management Systems, Addison-Wesley, Wokingham.

Wagner I., 1993, "A Web of Fuzzy Problems: Confronting the Ethical Issues", *Communications of the ACM, 36[4]*, pp 94 - 103.

Walker J. H., 1991, "Authoring Tools for Complex Document Sets", *Journal of Organisational Computing*, pp 132 - 147.

Williams G., 1987, "HyperCard", *BYTE, 14[12]*, pp 109 - 117.

Williams J. L., 1988, "Excelerator: A CASE Study", *Database Programming and Design, 1[4]*, pp 50 - 56.

Winkler D. and Kamins S., 1990, "HyperTalk 2.0: The Book", Bantam Books, U.S.A.

Winograd T., 1987, "A Language/Action Perspective on the Design of Cooperative Work", *Technical Report STAN-CS-87-1158*, Stanford Unversity Dept. of Computer Science.

Winsberg P., 1988, "CASE: Getting the Big Picture", *Database programming and Design, March*, pp 54 - 57.

Winter R., 1994, "Formalised Conceptual Models as a Foundation of Information Systems Development", *Proc. of the 13th International Conference on the Entity-Relationship Approach*, Manchester (U.K.), pp 437 - 455.

Wood J. R. G. and Wood-Harper A. T., 1993, "Information Technology in Support of Individual Decision Making", *Journal of Information Systems, 3*, pp 85 - 101.

Yakemovic K. C. B. and Conklin J. E., 1990, "Report on a Development Project Use of an Issue-Based Information System", *Proc. of CSCW '90*, pp 105 - 118.

Yankelovich N., Meyrowitz N. and van Dam A., 1985, "Reading and Writing the Electronic Notebook", *IEEE Computer, 18[10]*, pp 15 - 30.

Yao B. S., Waddle V. E. and Housel B. C., 1982, "View Modelling and Integration Using The Functional Data Model", *IEEE Transactions on Software Engineering, 8 [6]*, pp 544 - 553.

Yourdon E., 1993, "Decline and Fall of the American Programmer", Prentice-Hall, Engelwood Cliffs, N.J.

Zachman J., 1987, "A Framework for Information Systems Architecture", *IBM Systems Journal, 26[3]*, pp 276 - 292.